Section 7

DOCUMENTATION AND SOURCE LISTINGS

This section contains the documentation and source listings for the EISPACK subroutines and the documentation for the control program EISPAC. The subroutines appear in alphabetical order in Section 7.1 with the source listing for each subroutine following its document. Section 7.2 contains the document for EISPAC, but because of its size and machine dependencies the listing is not included.

Both the subroutine documents and source listings have been systematically edited for this publication to eliminate duplicate text; they therefore differ from the material distributed to requesters of EISPACK. The statement of certification of item 6 has been removed from the individual documents and appears here in Section 5. The reference to this publication which appears in item 4 and the brief discussion of testing that appears in items 5.A and 5.C of the distributed documents have been removed; the latter has been replaced by a pointer to the more extensive discussion of testing here in Section 3.

Two modifications have been made to the subroutine listings published here. First, they have been shortened by eliminating comments related to usage, calling sequence, and error exits which basically duplicate text in the documents. Second, statements initializing variables to machine-dependent constants, which in each distributed version of EISPACK are completed to contain the appropriate numerical values of these constants, appear here as

```
MACHEP = ?
RADIX = #
```

(The correct value for MACHEP is the smallest positive working precision floating point number which, when added to 1.0 using the working precision addition operation on your machine, gives a number larger than 1.0; that for RADIX is the floating point constant representing the base of the floating point arithmetic on your machine.) Appropriate values of MACHEP and RADIX for several machines are summarized in the following table.

MACHINE	MACHEP	RADIX
Burroughs 6700	2.**(-37)	8.
CDC 6000 and 7000 Series	2.**(-47)	2.
DEC PDP-10	2.**(-26)	2.
Honeywell 6070	2.**(-26)	2.
Univac 1110	2.**(-26)	2.

Except for the statements initializing MACHEP and RADIX, the published listings are ANSI Fortran. The double precision version certified for IBM equipment employs non-standard Fortran to the extent of using REAL*8 declarations and a small amount of complex double precision arithmetic. For this version, the appropriate values of MACHEP and RADIX are 16.D0**(-13) and 16.D0.

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F281-2 BAKVEC

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Symmetric Tridiagonal Matrix Determined by FIGI.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine BAKVEC forms the eigenvectors of a certain real non-symmetric tridiagonal matrix from the eigenvectors of that symmetric tridiagonal matrix determined by FIGI (F280).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE BAKVEC(NM,N,T,E,M,Z,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays T and Z as specified in the DIMENSION statements for T and Z in the calling program.

N is an integer input variable set equal to the order of the matrix T. N must be not greater than NM.

T is a working precision real input two-dimensional variable with row dimension NM and column dimension at least 3. T contains the non-symmetric tridiagonal matrix of order N in its first three columns. The subdiagonal elements are stored in the last N-1 positions of the

first column. The diagonal elements are stored in the second column. The superdiagonal elements are stored in the first N-1 positions of the third column. Elements T(1,1) and T(N,3) are arbitrary.

E is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary. Note that BAKVEC destroys E.

M is an integer input variable set equal to the number of columns of Z to be back transformed.

Z is a working precision real two-dimensional variable with row dimension NM and column dimension at least M. On input, the first M columns of Z contain the eigenvectors to be back transformed. On output, these columns of Z contain the transformed eigenvectors. The transformed eigenvectors are not normalized.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If E(I) is zero but either T(I,1) or T(I-1,3) is not zero, BAKVEC terminates with IERR set to 2*N+I and with no eigenvectors back transformed. In this case, the transformation by FIGI was not a similarity transformation.

If the above error condition does not occur, IERR is set to zero.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine FIGI (F280).

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the non-symmetric tridiagonal matrix T has been transformed to the tridiagonal symmetric matrix F by the similarity transformation

$$F = V^{-1} T V$$

where V is the diagonal matrix defined in FIGI (F280). Then, given an array Z of column vectors, BAKVEC computes the matrix product VZ. If the eigenvectors of F are columns of the array Z, then BAKVEC forms the eigenvectors of T in their place.

This subroutine is an implementation of the algorithm discussed in detail by Wilkinson (1).

4. REFERENCES.

1) Wilkinson, J.H., The Algebraic Eigenvalue Problem, Oxford, Clarendon Press, 335-337 (1965).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The accuracy of BAKVEC can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of certain real non-symmetric tridiagonal matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE BAKVEC(NM,N,T,E,M,Z,IERR)
C
      INTEGER I,J,M,N,NM,IERR
      REAL T(NM,3),E(N),Z(NM,M)
C
      IERR = 0
      IF (M .EQ. 0) GO TO 1001
      E(1) = 1.0
      IF (N .EQ. 1) GO TO 1001
C
      DO 100 I = 2, N
         IF (E(I) .NE. 0.0) GO TO 80
         IF (T(I,1) .NE. 0.0 .OR. T(I-1,3) .NE. 0.0) GO TO 1000
         E(I) = 1.0
         GO TO 100
   80    E(I) = E(I-1) * E(I) / T(I-1,3)
  100 CONTINUE
C
      DO 120 J = 1, M
C
         DO 120 I = 2, N
         Z(I,J) = Z(I,J) * E(I)
  120 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- EIGENVECTORS CANNOT BE
C                FOUND BY THIS PROGRAM **********
 1000 IERR = 2 * N + I
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F269-2 BALANC

A Fortran IV Subroutine to Balance a Real General Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine BALANC balances a real general matrix and isolates eigenvalues whenever possible. Sums of the magnitudes of elements in corresponding rows and columns are made nearly equal by exact similarity transformations, and eigenvalues are isolated by permutation similarity transformations. Balancing reduces the 1-norm of the original matrix whenever sums of the magnitudes of elements in some row and corresponding column are markedly different, while at the same time leaving the eigenvalues unchanged. Reducing the norm in this way can improve the accuracy of the computed eigenvalues and/or eigenvectors.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  BALANC(NM,N,A,LOW,IGH,SCALE)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array A as specified in the DIMENSION statement for A in the calling program.

N — is an integer input variable set equal to the order of the matrix A. N must be not greater than NM.

A is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, A contains the matrix of order N to be balanced. On output, A contains the transformed matrix.

LOW,IGH are integer output variables indicating the boundary indices for the balanced matrix. See section 3 for the details.

SCALE is a working precision real output one-dimensional variable of dimension at least N containing information about the similarity transformations. See section 3 for the details.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

It is particularly advantageous to use BALANC whenever the sums of the magnitudes of elements in some row and corresponding column are quite different. Additionally, the execution time for BALANC is small compared with the execution time for the routines which determine the eigenvalues and/or eigenvectors. Hence it is recommended that BALANC be used generally.

The subroutine BALBAK (F270) should be used to retrieve the eigenvectors of the original matrix after the eigenvectors of the transformed matrix have been determined.

3. DISCUSSION OF METHOD AND ALGORITHM.

First BALANC determines a product P of permutation matrices such that

$$PAP = \begin{pmatrix} T & X & Y \\ 0 & B & Z \\ 0 & 0 & R \end{pmatrix}$$

where T and R are upper triangular matrices and B is a square matrix situated in rows and columns LOW through IGH with no zero off-diagonal rows or columns. X, Y, and Z are rectangular matrices of appropriate dimensions. The

diagonal elements of T and R are the isolated eigenvalues of A. In the exceptional case where B is empty, LOW = 1 and IGH = 0 -- however, 1 is returned as the output value of IGH.

Next, the subroutine BALANC determines iteratively a non-singular diagonal matrix D of order IGH-LOW+1 such that

$$D^{-1} B D$$

is a balanced matrix in the sense that the absolute sums of the magnitudes of elements in corresponding rows and columns of

$$D^{-1} B D$$

are nearly equal. The diagonal elements of D are powers of the radix of the floating point arithmetic of the machine.

Upon completion of BALANC, the vector array SCALE contains the information about P and D necessary for BALBAK to retrieve the eigenvectors of the original matrix from the eigenvectors of the transformed matrix. The information is encoded as follows.

1. The J-th and SCALE(J)-th rows and columns of the original matrix have been permuted for J = 1,2,...,LOW-1,IGH+1,IGH+2,...,N.
2. SCALE(J) is the (J-LOW+1)-th diagonal element of D for J = LOW,LOW+1,...,IGH.

Upon completion of BALANC, the output matrix is

$$\begin{pmatrix} T & XD & Y \\ 0 & D^{-1} B D & D^{-1} Z \\ 0 & 0 & R \end{pmatrix}.$$

This subroutine is a translation of the Algol procedure BALANCE written and discussed in detail by Parlett and Reinsch (1).

4. REFERENCES.

1) Parlett, B.N. and Reinsch, C., Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors, Num. Math. 13,293-304 (1969). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/11, 315-326, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

BALANC introduces no rounding errors (on a machine with at least one guard digit) since its only arithmetic is scaling the input matrix by powers of the radix.

```
      SUBROUTINE BALANC(NM,N,A,LOW,IGH,SCALE)
C
      INTEGER I,J,K,L,M,N,JJ,NM,IGH,LOW,IEXC
      REAL A(NM,N),SCALE(N)
      REAL C,F,G,R,S,B2,RADIX
      REAL ABS
      LOGICAL NOCONV
C
C     ********** RADIX IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE BASE OF THE MACHINE FLOATING POINT REPRESENTATION.
C
C                **********
      RADIX = #
C
      B2 = RADIX * RADIX
      K = 1
      L = N
      GO TO 100
C     ********** IN-LINE PROCEDURE FOR ROW AND
C                COLUMN EXCHANGE **********
   20 SCALE(M) = J
      IF (J .EQ. M) GO TO 50
C
      DO 30 I = 1, L
         F = A(I,J)
         A(I,J) = A(I,M)
         A(I,M) = F
   30 CONTINUE
C
      DO 40 I = K, N
         F = A(J,I)
         A(J,I) = A(M,I)
         A(M,I) = F
   40 CONTINUE
C
   50 GO TO (80,130), IEXC
C     ********** SEARCH FOR ROWS ISOLATING AN EIGENVALUE
C                AND PUSH THEM DOWN **********
   80 IF (L .EQ. 1) GO TO 280
      L = L - 1
C     ********** FOR J=L STEP -1 UNTIL 1 DO -- **********
  100 DO 120 JJ = 1, L
         J = L + 1 - JJ
C
         DO 110 I = 1, L
            IF (I .EQ. J) GO TO 110
            IF (A(J,I) .NE. 0.0) GO TO 120
  110    CONTINUE
C
         M = L
         IEXC = 1
         GO TO 20
  120 CONTINUE
```

```
C
      GO TO 140
C     ********** SEARCH FOR COLUMNS ISOLATING AN EIGENVALUE
C                AND PUSH THEM LEFT **********
  130 K = K + 1
C
  140 DO 170 J = K, L
C
         DO 150 I = K, L
            IF (I .EQ. J) GO TO 150
            IF (A(I,J) .NE. 0.0) GO TO 170
  150    CONTINUE
C
         M = K
         IEXC = 2
         GO TO 20
  170 CONTINUE
C     ********** NOW BALANCE THE SUBMATRIX IN ROWS K TO L **********
      DO 180 I = K, L
  180 SCALE(I) = 1.0
C     ********** ITERATIVE LOOP FOR NORM REDUCTION **********
  190 NOCONV = .FALSE.
C
      DO 270 I = K, L
         C = 0.0
         R = 0.0
C
         DO 200 J = K, L
            IF (J .EQ. I) GO TO 200
            C = C + ABS(A(J,I))
            R = R + ABS(A(I,J))
  200    CONTINUE
C     ********** GUARD AGAINST ZERO C OR R DUE TO UNDERFLOW **********
         IF (C .EQ. 0.0 .OR. R .EQ. 0.0) GO TO 270
         G = R / RADIX
         F = 1.0
         S = C + R
  210    IF (C .GE. G) GO TO 220
         F = F * RADIX
         C = C * B2
         GO TO 210
  220    G = R * RADIX
  230    IF (C .LT. G) GO TO 240
         F = F / RADIX
         C = C / B2
         GO TO 230
C     ********** NOW BALANCE **********
  240    IF ((C + R) / F .GE. 0.95 * S) GO TO 270
         G = 1.0 / F
         SCALE(I) = SCALE(I) * F
         NOCONV = .TRUE.
C
         DO 250 J = K, N
  250    A(I,J) = A(I,J) * G
```

```
C
         DO 260 J = 1, L
  260    A(J,I) = A(J,I) * F
C
  270 CONTINUE
C
      IF (NOCONV) GO TO 190
C
  280 LOW = K
      IGH = L
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F270-2 BALBAK

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Real Matrix Transformed by BALANC.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine BALBAK forms the eigenvectors of a real general matrix from the eigenvectors of that matrix transformed by BALANC (F269).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  BALBAK(NM,N,LOW,IGH,SCALE,M,Z)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array Z as specified in the DIMENSION statement for Z in the calling program.

N — is an integer input variable set equal to the number of components of the vectors in the array Z. N must be not greater than NM.

LOW,IGH — are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F269 for the details.

SCALE — is a working precision real input one-dimensional variable of dimension at least N containing information about the transformations. See section 3 of F269 for the details.

M is an integer input variable set equal to the number of columns of Z to be back transformed.

Z is a working precision real two-dimensional variable with row dimension NM and column dimension at least M. On input, the first M columns of Z contain the real and imaginary parts of the eigenvectors to be back transformed. On output, these M columns of Z contain the real and imaginary parts of the transformed eigenvectors.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine BALANC (F269).

3. DISCUSSION OF METHOD AND ALGORITHM.

Using the notation of F269, the original matrix A is transformed into the matrix C by the similarity transformation

$$C = G^{-1} PAPG$$

where G is the diagonal matrix whose diagonal elements are 1.0 in positions 1 to LOW-1, 1.0 in positions IGH+1 to N, and are the diagonal elements of D in positions LOW to IGH. Given an array Z of vectors, the subroutine BALBAK computes the matrix product PGZ. If the real and imaginary parts of the eigenvectors of C are columns of the array Z, then BALBAK forms the eigenvectors of A in their place. The information about P and D is encoded in the array SCALE. See section 3 of F269 for the details.

This subroutine is a translation of the Algol procedure BALBAK written and discussed in detail by Parlett and Reinsch (1).

4. REFERENCES.

1) Parlett, B.N. and Reinsch, C., Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors, Num. Math. 13,293-304 (1969). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/11, 315-326, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

BALBAK introduces no rounding errors (on a machine with at least one guard digit) since its only arithmetic is scaling the eigenvector matrix by powers of the radix.

```
      SUBROUTINE BALBAK(NM,N,LOW,IGH,SCALE,M,Z)
C
      INTEGER I,J,K,M,N,II,NM,IGH,LOW
      REAL SCALE(N),Z(NM,M)
      REAL S
C
      IF (M .EQ. 0) GO TO 200
      IF (IGH .EQ. LOW) GO TO 120
C
      DO 110 I = LOW, IGH
         S = SCALE(I)
C     ********** LEFT HAND EIGENVECTORS ARE BACK TRANSFORMED
C                IF THE FOREGOING STATEMENT IS REPLACED BY
C                S=1.0/SCALE(I). **********
         DO 100 J = 1, M
  100    Z(I,J) = Z(I,J) * S
C
  110 CONTINUE
C     *********- FOR I=LOW-1 STEP -1 UNTIL 1,
C               IGH+1 STEP 1 UNTIL N DO -- **********
  120 DO 140 II = 1, N
         I = II
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 140
         IF (I .LT. LOW) I = LOW - II
         K = SCALE(I)
         IF (K .EQ. I) GO TO 140
C
         DO 130 J = 1, M
            S = Z(I,J)
            Z(I,J) = Z(K,J)
            Z(K,J) = S
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F294 BISECT

A Fortran IV Subroutine to Determine Some Eigenvalues of a Symmetric Tridiagonal Matrix.

May, 1972

1. PURPOSE.

The Fortran IV subroutine BISECT determines those eigenvalues of a symmetric tridiagonal matrix in a specified interval using Sturm sequencing.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  BISECT(N,EPS1,D,E,E2,LB,UB,
                   MM,M,W,IND,IERR,RV4,RV5)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

N — is an integer input variable set equal to the order of the matrix.

EPS1 — is a working precision real variable. On input, it specifies an absolute error tolerance for the computed eigenvalues. If the input EPS1 is non-positive, it is reset to a default value described in section 2C.

D — is a working precision real input one-dimensional variable of dimension at least N containing the diagonal elements of the symmetric tridiagonal matrix.

E — is a working precision real input one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary.

E2 is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the squares of the subdiagonal elements of the symmetric tridiagonal matrix. E2(1) is arbitrary. On output, E2(1) is set to zero. If any of the elements in E are regarded as negligible, the corresponding elements of E2 are set to zero, and so the matrix splits into a direct sum of submatrices.

LB,UB are working precision real input variables specifying the lower and upper endpoints, respectively, of the interval to be searched for the eigenvalues. If LB is not less than UB, BISECT computes no eigenvalues. See section 2C for further details.

MM is an integer input variable set equal to an upper bound for the number of eigenvalues in the interval (LB,UB).

M is an integer output variable set equal to the number of eigenvalues determined to lie in the interval (LB,UB).

W is a working precision real output one-dimensional variable of dimension at least MM containing the M eigenvalues of the symmetric tridiagonal matrix in the interval (LB,UB). The eigenvalues are in ascending order in W.

IND is an integer output one-dimensional variable of dimension at least MM containing the submatrix indices associated with the corresponding M eigenvalues in W. Eigenvalues belonging to the first submatrix have index 1, those belonging to the second submatrix have index 2, etc.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

RV4,RV5 are working precision real temporary one-dimensional variables of dimension at least N used to hold the lower and upper bounds for the eigenvalues in the bisection process.

B. Error Conditions and Returns.

If M exceeds MM, BISECT terminates with no eigenvalues computed, and IERR is set to 3*N+1. Upon this error exit, M contains the number of eigenvalues determined to lie in (LB,UB).

If M does not exceed MM, IERR is set to zero.

C. Applicability and Restrictions.

To determine some of the eigenvalues of a full symmetric matrix, BISECT should be preceded by TRED1 (F277) to provide a suitable symmetric tridiagonal matrix for BISECT.

To determine some of the eigenvalues of a complex Hermitian matrix, BISECT should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for BISECT.

Some of the eigenvalues of certain non-symmetric tridiagonal matrices can be computed using the combination of FIGI (F280) and BISECT. See F280 for the description of this special class of matrices. For these matrices, BISECT should be preceded by FIGI to provide a suitable symmetric matrix for BISECT.

To determine eigenvectors associated with the computed eigenvalues, BISECT should be followed by TINVIT (F223) and the appropriate back transformation subroutine -- TRBAK1 (F279) after TRED1, HTRIBK (F285) after HTRIDI, or BAKVEC (F281) after FIGI.

The subroutines TQL1 (F289) and IMTQL1 (F291) determine all the eigenvalues of a symmetric tridiagonal matrix faster than BISECT determines 25 percent of them. Hence, if more than 25 percent of them are desired, it is recommended that TQL1 or IMTQL1 be used.

The interval (LB,UB) is formally half-open, not including the upper endpoint UB. However, because of rounding errors, the true eigenvalues very close to the endpoints of the interval may be erroneously counted or missed.

The input interval (LB,UB) may be refined internally to a smaller interval known to contain all the eigenvalues in (LB,UB). This insures that BISECT will not perform unnecessary bisection steps to determine the eigenvalues in (LB,UB).

The precision of the computed eigenvalues is controlled through the parameter EPS1. To obtain eigenvalues accurate to within a certain absolute error, EPS1 should be set to that error. In particular, if MACHEP denotes the relative machine precision, then to obtain the eigenvalues to an accuracy commensurate with small relative perturbations of the order of MACHEP in the matrix elements, it is enough for most tridiagonal matrices that EPS1 be approximately MACHEP times a norm of the matrix. But some matrices require a smaller EPS1 for this accuracy, perhaps as small as MACHEP times the eigenvalue of smallest magnitude in (LB,UB). Note, however, that if EPS1 is smaller than required, BISECT will perform unnecessary bisection steps to determine the eigenvalues. For further discussion of EPS1, see reference (1).

If the input EPS1 is non-positive, BISECT resets it, for each submatrix, to -MACHEP times the 1-norm of the submatrix and uses its magnitude. This value is tentatively considered adequate for computing the eigenvalues to an accuracy commensurate with small relative perturbations of the order of MACHEP in the matrix elements.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the method of bisection applied to the Sturm sequence.

The calculations proceed as follows. First, the subdiagonal elements are tested for negligibility. If an element is considered negligible, its square is set to zero, and so the matrix splits into a direct sum of submatrices. Then, the Sturm sequence for the entire matrix is evaluated at UB and LB giving the number of eigenvalues of the matrix less than UB and LB respectively. The difference is the number of eigenvalues in (LB,UB).

Next, a submatrix is examined for its eigenvalues in the interval (LB,UB). The Gerschgorin interval, known to contain all the eigenvalues, is determined and used to refine the input interval (LB,UB). If the input EPS1 is non-positive, it is reset to the default value described in section 2C. Then, subintervals, each enclosing an eigenvalue in (LB,UB), are shrunk using a bisection process until the endpoints of each subinterval are close enough to be accepted as an eigenvalue of the submatrix. Here the endpoints of each subinterval are close enough when they differ by less than MACHEP times twice the sum of the magnitudes of the endpoints plus the absolute error tolerance EPS1.

The submatrix eigenvalues are then merged with previously found eigenvalues into an ordered set.

The above steps are repeated on each submatrix until all the eigenvalues in the interval (LB,UB) are computed.

This subroutine is a subset (except for the section merging eigenvalues of submatrices) of the Fortran subroutine TSTURM (F293), which is a translation of the Algol procedure TRISTURM written and discussed in detail by Peters and Wilkinson (2). A similar Algol procedure BISECT is discussed in detail by Barth, Martin, and Wilkinson (1).

4. REFERENCES.

1) Barth, W., Martin, R.S., and Wilkinson, J.H., Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection, Num. Math. 9,386-393 (1967).

2) Peters, G. and Wilkinson, J.H., The Calculation of Specified Eigenvectors by Inverse Iteration, Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/18, 418-439, Springer-Verlag, 1971.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine BISECT is numerically stable (1,2); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix.

```
      SUBROUTINE BISECT(N,EPS1,D,E,E2,LB,UB,MM,M,W,IND,IERR,RV4,RV5)
C
      INTEGER I,J,K,L,M,N,P,Q,R,S,II,MM,M1,M2,TAG,IERR,ISTURM
      REAL D(N),E(N),E2(N),W(MM),RV4(N),RV5(N)
      REAL U,V,LB,T1,T2,UB,XU,X0,X1,EPS1,MACHEP
      REAL ABS,AMAX1,AMIN1,FLOAT
      INTEGER IND(MM)
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      TAG = 0
      T1 = LB
      T2 = UB
C     ********** LOOK FOR SMALL SUB-DIAGONAL ENTRIES **********
      DO 40 I = 1, N
         IF (I .EQ. 1) GO TO 20
         IF (ABS(E(I)) .GT. MACHEP * (ABS(D(I)) + ABS(D(I-1))))
     X      GO TO 40
   20    E2(I) = 0.0
   40 CONTINUE
C     ********** DETERMINE THE NUMBER OF EIGENVALUES
C                IN THE INTERVAL **********
      P = 1
      Q = N
      X1 = UB
      ISTURM = 1
      GO TO 320
   60 M = S
      X1 = LB
      ISTURM = 2
      GO TO 320
   80 M = M - S
      IF (M .GT. MM) GO TO 980
      Q = 0
      R = 0
C     ********** ESTABLISH AND PROCESS NEXT SUBMATRIX, REFINING
C                INTERVAL BY THE GERSCHGORIN BOUNDS **********
  100 IF (R .EQ. M) GO TO 1001
      TAG = TAG + 1
      P = Q + 1
      XU = D(P)
      X0 = D(P)
      U = 0.0
```

```
C
      DO 120 Q = P, N
         X1 = U
         U = 0.0
         V = 0.0
         IF (Q .EQ. N) GO TO 110
         U = ABS(E(Q+1))
         V = E2(Q+1)
  110    XU = AMIN1(D(Q)-(X1+U),XU)
         X0 = AMAX1(D(Q)+(X1+U),X0)
         IF (V .EQ. 0.0) GO TO 140
  120 CONTINUE
C
  140 X1 = AMAX1(ABS(XU),ABS(X0)) * MACHEP
      IF (EPS1 .LE. 0.0) EPS1 = -X1
      IF (P .NE. Q) GO TO 180
C     ********** CHECK FOR ISOLATED ROOT WITHIN INTERVAL **********
      IF (T1 .GT. D(P) .OR. D(P) .GE. T2) GO TO 940
      M1 = P
      M2 = P
      RV5(P) = D(P)
      GO TO 900
  180 X1 = X1 * FLOAT(Q-P+1)
      LB = AMAX1(T1,XU-X1)
      UB = AMIN1(T2,X0+X1)
      X1 = LB
      ISTURM = 3
      GO TO 320
  200 M1 = S + 1
      X1 = UB
      ISTURM = 4
      GO TO 320
  220 M2 = S
      IF (M1 .GT. M2) GO TO 940
C     ********** FIND ROOTS BY BISECTION **********
      X0 = UB
      ISTURM = 5
C
      DO 240 I = M1, M2
         RV5(I) = UB
         RV4(I) = LB
  240 CONTINUE
C     ********** LOOP FOR K-TH EIGENVALUE
C                 FOR K=M2 STEP -1 UNTIL M1 DO --
C                 (-DO- NOT USED TO LEGALIZE COMPUTEE-GO-TO) **********
      K = M2
  250    XU = LB
C     ********** FOR I=K STEP -1 UNTIL M1 DO -- **********
         DO 260 II = M1, K
            I = M1 + K - II
            IF (XU .GE. RV4(I)) GO TO 260
            XU = RV4(I)
            GO TO 280
  260    CONTINUE
```

```
C
  280     IF (X0 .GT. RV5(K)) X0 = RV5(K)
C      ********** NEXT BISECTION STEP **********
  300     X1 = (XU + X0) * 0.5
          IF ((X0 - XU) .LE. (2.0 * MACHEP *
     X       (ABS(XU) + ABS(X0)) + ABS(EPS1))) GO TO 420
C      ********** IN-LINE PROCEDURE FOR STURM SEQUENCE **********
  320     S = P - 1
          U = 1.0
C
          DO 340 I = P, Q
             IF (U .NE. 0.0) GO TO 325
             V = ABS(E(I)) / MACHEP
             GO TO 330
  325        V = E2(I) / U
  330        U = D(I) - X1 - V
             IF (U .LT. 0.0) S = S + 1
  340     CONTINUE
C
          GO TO (60,80,200,220,360), ISTURM
C      ********** REFINE INTERVALS **********
  360     IF (S .GE. K) GO TO 400
          XU = X1
          IF (S .GE. M1) GO TO 380
          RV4(M1) = X1
          GO TO 300
  380     RV4(S+1) = X1
          IF (RV5(S) .GT. X1) RV5(S) = X1
          GO TO 300
  400     X0 = X1
          GO TO 300
C      ********** K-TH EIGENVALUE FOUND **********
  420     RV5(K) = X1
      K = K - 1
      IF (K .GE. M1) GO TO 250
C      ********** ORDER EIGENVALUES TAGGED WITH THEIR
C                 SUBMATRIX ASSOCIATIONS **********
  900 S = R
      R = R + M2 - M1 + 1
      J = 1
      K = M1
C
      DO 920 L = 1, R
         IF (J .GT. S) GO TO 910
         IF (K .GT. M2) GO TO 940
         IF (RV5(K) .GE. W(L)) GO TO 915
C
         DO 905 II = J, S
            I = L + S - II
            W(I+1) = W(I)
            IND(I+1) = IND(I)
  905    CONTINUE
```

```
C
  910     W(L) = RV5(K)
          IND(L) = TAG
          K = K + 1
          GO TO 920
  915     J = J + 1
  920 CONTINUE
C
  940 IF (Q .LT. N) GO TO 100
      GO TO 1001
C     ********** SET ERROR -- UNDERESTIMATE OF NUMBER OF
C                EIGENVALUES IN INTERVAL **********
  980 IERR = 3 * N + 1
 1001 LB = T1
      UB = T2
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F272-2 CBABK2

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Complex Matrix Transformed by CBAL.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine CBABK2 forms the eigenvectors of a complex general matrix from the eigenvectors of that matrix transformed by CBAL (F271).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE CBABK2(NM,N,LOW,IGH,SCALE,M,ZR,ZI)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays ZR and ZI as specified in the DIMENSION statements for ZR and ZI in the calling program.

N is an integer input variable set equal to the number of components of the vectors in the array Z = (ZR,ZI). N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F271 for the details.

SCALE is a working precision real input one-dimensional variable of dimension at least N containing information about the transformations. See section 3 of F271 for the details.

M is an integer input variable set equal to the number of columns of Z = (ZR,ZI) to be back transformed.

ZR,ZI are working precision real two-dimensional variables with row dimension NM and column dimension at least M. On input, the first M columns of ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors to be back transformed. On output, these M columns of ZR and ZI contain the real and imaginary parts of the transformed eigenvectors.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine CBAL (F271).

3. DISCUSSION OF METHOD AND ALGORITHM.

Using the notation of F271, the original matrix A is transformed into the matrix C by the similarity transformation

$$C = G^{-1} PAPG$$

where G is the diagonal matrix whose diagonal elements are 1.0 in positions 1 to LOW-1, 1.0 in positions IGH+1 to N, and are the diagonal elements of D in positions LOW to IGH. Given an array Z of vectors, the subroutine CBABK2 computes the matrix product PGZ. If the eigenvectors of C are columns of the array Z, then CBABK2 forms the eigenvectors of A in their place. The information about P and D is encoded in the array SCALE. See section 3 of F271 for the details.

This subroutine is a translation of the Algol procedure CBABK2 which is the analogue for complex matrices of the procedure BALBAK written and discussed in detail by Parlett and Reinsch (1).

4. REFERENCES.

1) Parlett, B.N. and Reinsch, C., Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors, Num. Math. 13,293-304 (1969). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/11, 315-326, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

CBABK2 introduces no rounding errors (on a machine with at least one guard digit) since its only arithmetic is scaling the eigenvector matrix by powers of the radix.

```
      SUBROUTINE CBABK2(NM,N,LOW,IGH,SCALE,M,ZR,ZI)
C
      INTEGER I,J,K,M,N,II,NM,IGH,LOW
      REAL SCALE(N),ZR(NM,M),ZI(NM,M)
      REAL S
C
      IF (M .EQ. 0) GO TO 200
      IF (IGH .EQ. LOW) GO TO 120
C
      DO 110 I = LOW, IGH
         S = SCALE(I)
C     ********** LEFT HAND EIGENVECTORS ARE BACK TRANSFORMED
C                IF THE FOREGOING STATEMENT IS REPLACED BY
C                S=1.0/SCALE(I). **********
         DO 100 J = 1, M
            ZR(I,J) = ZR(I,J) * S
            ZI(I,J) = ZI(I,J) * S
  100    CONTINUE
C
  110 CONTINUE
C     ********** FOR I=LOW-1 STEP -1 UNTIL 1,
C                IGH+1 STEP 1 UNTIL N DO -- **********
  120 DO 140 II = 1, N
         I = II
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 140
         IF (I .LT. LOW) I = LOW - II
         K = SCALE(I)
         IF (K .EQ. I) GO TO 140
C
         DO 130 J = 1, M
            S = ZR(I,J)
            ZR(I,J) = ZR(K,J)
            ZR(K,J) = S
            S = ZI(I,J)
            ZI(I,J) = ZI(K,J)
            ZI(K,J) = S
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F271-2 CBAL

A Fortran IV Subroutine to Balance a Complex General Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine CBAL balances a complex general matrix and isolates eigenvalues whenever possible. Sums of the magnitudes of elements in corresponding rows and columns are made nearly equal by exact similarity transformations, and eigenvalues are isolated by permutation similarity transformations. Balancing reduces the 1-norm of the original matrix whenever sums of the magnitudes of elements in some row and corresponding column are markedly different, while at the same time leaving the eigenvalues unchanged. Reducing the norm in this way can improve the accuracy of the computed eigenvalues and/or eigenvectors.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  CBAL(NM,N,AR,AI,LOW,IGH,SCALE)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays AR and AI as specified in the DIMENSION statements for AR and AI in the calling program.

N — is an integer input variable set equal to the order of the matrix A = (AR,AI). N must be not greater than NM.

AR,AI are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, AR and AI contain the real and imaginary parts, respectively, of the complex matrix of order N to be balanced. On output, AR and AI contain the real and imaginary parts of the transformed matrix.

LOW,IGH are integer output variables indicating the boundary indices for the balanced matrix. See section 3 for the details.

SCALE is a working precision real output one-dimensional variable of dimension at least N containing information about the similarity transformations. See section 3 for the details.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

It is particularly advantageous to use CBAL whenever the sums of the magnitudes of elements in some row and corresponding column are quite different. Additionally, the execution time for CBAL is small compared with the execution time for the routines which determine the eigenvalues and/or eigenvectors. Hence it is recommended that CBAL be used generally.

The subroutine CBABK2 (F272) should be used to retrieve the eigenvectors of the original matrix after the eigenvectors of the transformed matrix have been determined.

In this implementation, the arithmetic is real throughout.

3. DISCUSSION OF METHOD AND ALGORITHM.

First CBAL determines a product P of permutation matrices such that

$$PAP = \begin{pmatrix} T & X & Y \\ 0 & B & Z \\ 0 & 0 & R \end{pmatrix}$$

where T and R are upper triangular matrices and B is a square matrix situated in rows and columns LOW through IGH with no zero off-diagonal rows or columns. X, Y, and Z are rectangular matrices of appropriate dimensions. The diagonal elements of T and R are the isolated eigenvalues of A. In the exceptional case where B is empty, LOW = 1 and IGH = 0 -- however, 1 is returned as the output value of IGH.

Next, the subroutine CBAL determines iteratively a real non-singular diagonal matrix D of order IGH-LOW+1 such that

$$D^{-1}BD$$

is a balanced matrix in the sense that the sums of the magnitudes of elements in corresponding rows and columns of

$$D^{-1}BD$$

are nearly equal. The diagonal elements of D are powers of the radix of the floating point arithmetic of the machine.

Upon completion of CBAL, the vector array SCALE contains the information about P and D necessary for CBABK2 to retrieve the eigenvectors of the original matrix from the eigenvectors of the transformed matrix. The information is encoded as follows.

1. The J-th and SCALE(J)-th rows and columns of the original matrix have been permuted for J = 1,2,...,LOW-1,IGH+1,IGH+2,...,N.
2. SCALE(J) is the (J-LOW+1)-th diagonal element of D for J = LOW,LOW+1,...,IGH.

Upon completion of CBAL, the output matrix is

$$\begin{pmatrix} T & XD & Y \\ 0 & D^{-1}BD & D^{-1}Z \\ 0 & 0 & R \end{pmatrix}.$$

This subroutine is a translation of the Algol procedure CBALANCE which is the analogue for complex matrices of the procedure BALANCE written and discussed in detail by Parlett and Reinsch (1).

4. REFERENCES.

1) Parlett, B.N. and Reinsch, C., Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors, Num. Math. 13,293-304 (1969). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/11, 315-326, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

CBAL introduces no rounding errors (on a machine with at least one guard digit) since its only arithmetic is scaling the input matrix by powers of the radix.

```
      SUBROUTINE CBAL(NM,N,AR,AI,LOW,IGH,SCALE)
C
      INTEGER I,J,K,L,M,N,JJ,NM,IGH,LOW,IEXC
      REAL AR(NM,N),AI(NM,N),SCALE(N)
      REAL C,F,G,R,S,B2,RADIX
      REAL ABS
      LOGICAL NOCONV
C
C     ********** RADIX IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE BASE OF THE MACHINE FLOATING POINT REPRESENTATION.
C
C                **********
      RADIX = #
C
      B2 = RADIX * RADIX
      K = 1
      L = N
      GO TO 100
C     ********** IN-LINE PROCEDURE FOR ROW AND
C                COLUMN EXCHANGE **********
   20 SCALE(M) = J
      IF (J .EQ. M) GO TO 50
C
      DO 30 I = 1, L
         F = AR(I,J)
         AR(I,J) = AR(I,M)
         AR(I,M) = F
         F = AI(I,J)
         AI(I,J) = AI(I,M)
         AI(I,M) = F
   30 CONTINUE
C
      DO 40 I = K, N
         F = AR(J,I)
         AR(J,I) = AR(M,I)
         AR(M,I) = F
         F = AI(J,I)
         AI(J,I) = AI(M,I)
         AI(M,I) = F
   40 CONTINUE
C
   50 GO TO (80,130), IEXC
C     ********** SEARCH FOR ROWS ISOLATING AN EIGENVALUE
C                AND PUSH THEM DOWN **********
   80 IF (L .EQ. 1) GO TO 280
      L = L - 1
C     ********** FOR J=L STEP -1 UNTIL 1 DO -- **********
  100 DO 120 JJ = 1, L
         J = L + 1 - JJ
C
         DO 110 I = 1, L
            IF (I .EQ. J) GO TO 110
            IF (AR(J,I) .NE. 0.0 .OR. AI(J,I) .NE. 0.0) GO TO 120
  110    CONTINUE
```

```
C
         M = L
         IEXC = 1
         GO TO 20
  120 CONTINUE
C
      GO TO 140
C     ********** SEARCH FOR COLUMNS ISOLATING AN EIGENVALUE
C                AND PUSH THEM LEFT **********
  130 K = K + 1
C
  140 DO 170 J = K, L
C
         DO 150 I = K, L
            IF (I .EQ. J) GO TO 150
            IF (AR(I,J) .NE. 0.0 .OR. AI(I,J) .NE. 0.0) GO TO 170
  150    CONTINUE
C
         M = K
         IEXC = 2
         GO TO 20
  170 CONTINUE
C     ********** NOW BALANCE THE SUBMATRIX IN ROWS K TO L **********
      DO 180 I = K, L
  180 SCALE(I) = 1.0
C     ********** ITERATIVE LOOP FOR NORM REDUCTION **********
  190 NOCONV = .FALSE.
C
      DO 270 I = K, L
         C = 0.0
         R = 0.0
C
         DO 200 J = K, L
            IF (J .EQ. I) GO TO 200
            C = C + ABS(AR(J,I)) + ABS(AI(J,I))
            R = R + ABS(AR(I,J)) + ABS(AI(I,J))
  200    CONTINUE
C     ********** GUARD AGAINST ZERO C OR R DUE TO UNDERFLOW **********
         IF (C .EQ. 0.0 .OR. R .EQ. 0.0) GO TO 270
         G = R / RADIX
         F = 1.0
         S = C + R
  210    IF (C .GE. G) GO TO 220
         F = F * RADIX
         C = C * B2
         GO TO 210
  220    G = R * RADIX
  230    IF (C .LT. G) GO TO 240
         F = F / RADIX
         C = C / B2
         GO TO 230
```

```
C     ********** NOW BALANCE **********
  240    IF ((C + R) / F .GE. 0.95 * S) GO TO 270
         G = 1.0 / F
         SCALE(I) = SCALE(I) * F
         NOCONV = .TRUE.
C
         DO 250 J = K, N
            AR(I,J) = AR(I,J) * G
            AI(I,J) = AI(I,J) * G
  250    CONTINUE
C
         DO 260 J = 1, L
            AR(J,I) = AR(J,I) * F
            AI(J,I) = AI(J,I) * F
  260    CONTINUE
C
  270 CONTINUE
C
      IF (NOCONV) GO TO 190
C
  280 LOW = K
      IGH = L
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F301 CG

A Fortran IV Driver Subroutine to Determine the Eigenvalues and Eigenvectors of a Complex General Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine CG calls the recommended sequence of subroutines from the eigensystem subroutine package EISPACK to determine the eigenvalues and eigenvectors (if desired) of a complex general matrix.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  CG(NM,N,AR,AI,WR,WI,MATZ,
                ZR,ZI,FV1,FV2,FV3,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays AR, AI, ZR, and ZI as specified in the DIMENSION statements for AR, AI, ZR, and ZI in the calling program.

N — is an integer input variable set equal to the order of the matrix A = (AR,AI). N must not be greater than NM.

AR,AI — are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, AR and AI contain the real and imaginary parts, respectively, of the complex general matrix of order N whose eigenvalues and eigenvectors are to be found. On output, AR and AI have been destroyed.

WR,WI are working precision real output one-dimensional variables of dimension at least N containing the real and imaginary parts, respectively, of the eigenvalues of the complex general matrix.

MATZ is an integer input variable set equal to zero if only eigenvalues are desired; otherwise it is set to any non-zero integer for both eigenvalues and eigenvectors.

ZR,ZI are, if MATZ is non-zero, working precision real output two-dimensional variables with row dimension NM and column dimension at least N containing the real and imaginary parts, respectively, of the eigenvectors. The eigenvectors are not normalized. If MATZ is zero, ZR and ZI are not referenced and can be dummy variables.

FV1,FV2,FV3
are working precision temporary one-dimensional variables of dimension at least N.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If N is greater than NM, the subroutine terminates with IERR set equal to 10*N.

If more than 30 iterations are required to determine an eigenvalue, the subroutine terminates with IERR set equal to the index of the eigenvalue for which the failure occurs. The eigenvalues in the WR and WI arrays should be correct for indices IERR+1,IERR+2,...,N, but no eigenvectors are computed.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

This subroutine can be used to find all the eigenvalues and all the eigenvectors (if desired) of a complex general matrix.

3. DISCUSSION OF METHOD AND ALGORITHM.

This subroutine calls the recommended sequence of subroutines from EISPACK to find the eigenvalues and eigenvectors of a complex general matrix.

To find eigenvalues only, the sequence is the following.

- CBAL - to balance a complex general matrix.
- CORTH - to reduce the balanced matrix to an upper Hessenberg matrix using unitary transformations.
- COMQR - to determine the eigenvalues of the original matrix from the Hessenberg matrix.

To find eigenvalues and eigenvectors, the sequence is the following.

- CBAL - to balance a complex general matrix.
- CORTH - to reduce the balanced matrix to an upper Hessenberg matrix using unitary transformations.
- COMQR2 - to determine the eigenvalues and eigenvectors of the balanced matrix from the Hessenberg matrix.
- CBABK2 - to backtransform the eigenvectors to those of the original matrix.

4. REFERENCES.

1) Garbow, B.S. and Dongarra, J.J., EISPACK Path Chart, April, 1974.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The accuracy of CG can be best described in terms of its role in those paths of EISPACK (1) which find eigenvalues and eigenvectors of complex general matrices. In these paths, this subroutine is numerically stable. This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

```
      SUBROUTINE CG(NM,N,AR,AI,WR,WI,MATZ,ZR,ZI,FV1,FV2,FV3,IERR)
C
      INTEGER N,NM,IS1,IS2,IERR,MATZ
      REAL AR(NM,N),AI(NM,N),WR(N),WI(N),ZR(NM,N),ZI(NM,N),
     $       FV1(N),FV2(N),FV3(N)
C
      IF (N .LE. NM) GO TO 10
      IERR = 10 * N
      GO TO 50
C
   10 CALL  CBAL(NM,N,AR,AI,IS1,IS2,FV1)
      CALL  CORTH(NM,N,IS1,IS2,AR,AI,FV2,FV3)
      IF (MATZ .NE. 0) GO TO 20
C     ********** FIND EIGENVALUES ONLY **********
      CALL  COMQR(NM,N,IS1,IS2,AR,AI,WR,WI,IERR)
      GO TO 50
C     ********** FIND BOTH EIGENVALUES AND EIGENVECTORS **********
   20 CALL  COMQR2(NM,N,IS1,IS2,FV2,FV3,AR,AI,WR,WI,ZR,ZI,IERR)
      IF (IERR .NE. 0) GO TO 50
      CALL  CBABK2(NM,N,IS1,IS2,FV1,N,ZR,ZI)
   50 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F302 CH

A Fortran IV Driver Subroutine to Determine the Eigenvalues and Eigenvectors of a Complex Hermitian Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine CH calls the recommended sequence of subroutines from the eigensystem subroutine package EISPACK to determine the eigenvalues and eigenvectors (if desired) of a complex Hermitian matrix.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  CH(NM,N,AR,AI,W,MATZ,
                ZR,ZI,FV1,FV2,FM1,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays AR, AI, ZR, and ZI as specified in the DIMENSION statements for AR, AI, ZR, and ZI in the calling program.

N — is an integer input variable set equal to the order of the matrix A = (AR,AI). N must not be greater than NM.

AR,AI — are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, AR and AI contain the real and imaginary parts, respectively, of the complex Hermitian matrix of order N whose eigenvalues and

eigenvectors are to be found. Only the full lower triangle of AR and the strict lower triangle of AI need be supplied. On output, the full upper triangle of AR and the strict upper triangle of AI are unaltered.

W is a working precision real output one-dimensional variable of dimension at least N containing the eigenvalues of the complex Hermitian matrix in ascending order.

MATZ is an integer input variable set equal to zero if only eigenvalues are desired; otherwise it is set to any non-zero integer for both eigenvalues and eigenvectors.

ZR,ZI are, if MATZ is non-zero, working precision real output two-dimensional variables with row dimension NM and column dimension at least N containing the real and imaginary parts, respectively, of the eigenvectors. The eigenvectors are orthonormal. If MATZ is zero, ZR and ZI are not referenced and can be dummy variables.

FV1,FV2 are working precision temporary one-dimensional variables of dimension at least N.

FM1 is a working precision temporary two-dimensional variable with row dimension 2 and column dimension at least N.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If N is greater than NM, the subroutine terminates with IERR set equal to 10*N.

If more than 30 iterations are required to determine an eigenvalue, the subroutine terminates with IERR set equal to the index of the eigenvalue for which the failure occurs. The eigenvalues in the W array should be correct for indices 1,2,...,IERR-1, but no eigenvectors are computed.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

This subroutine can be used to find all the eigenvalues and all the eigenvectors (if desired) of a complex Hermitian matrix.

3. DISCUSSION OF METHOD AND ALGORITHM.

This subroutine calls the recommended sequence of subroutines from EISPACK to find the eigenvalues and eigenvectors of a complex Hermitian matrix.

To find eigenvalues only, the sequence is the following.

HTRIDI - to reduce a complex Hermitian matrix to a real symmetric tridiagonal matrix using unitary transformations.

TQLRAT - to determine the eigenvalues of the original matrix from the real symmetric tridiagonal matrix.

To find eigenvalues and eigenvectors, the sequence is the following.

HTRIDI - to reduce a complex Hermitian matrix to a real symmetric tridiagonal matrix using unitary transformations.

TQL2 - to determine the eigenvalues and eigenvectors of the real symmetric tridiagonal matrix.

HTRIBK - to backtransform the eigenvectors to those of the original matrix.

4. REFERENCES.

1) Garbow, B.S. and Dongarra, J.J., EISPACK Path Chart, April, 1974.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian matrices.

B. Accuracy.

The accuracy of CH can best be described in terms of its role in those paths of EISPACK (1) which find eigenvalues and eigenvectors of complex Hermitian matrices. In these paths, this subroutine is numerically stable. This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE CH(NM,N,AR,AI,W,MATZ,ZR,ZI,FV1,FV2,FM1,IERR)
C
      INTEGER I,J,N,NM,IERR,MATZ
      REAL AR(NM,N),AI(NM,N),W(N),ZR(NM,N),ZI(NM,N),
     $       FV1(N),FV2(N),FM1(2,N)
C
      IF (N .LE. NM) GO TO 10
      IERR = 10 * N
      GO TO 50
C
   10 CALL  HTRIDI(NM,N,AR,AI,W,FV1,FV2,FM1)
      IF (MATZ .NE. 0) GO TO 20
C     ********** FIND EIGENVALUES ONLY **********
      CALL  TQLRAT(N,W,FV2,IERR)
      GO TO 50
C     ********** FIND BOTH EIGENVALUES AND EIGENVECTORS **********
   20 DO 40 I = 1, N
C
         DO 30 J = 1, N
            ZR(J,I) = 0.0
   30    CONTINUE
C
         ZR(I,I) = 1.0
   40 CONTINUE
C
      CALL  TQL2(NM,N,W,FV1,ZR,IERR)
      IF (IERR .NE. 0) GO TO 50
      CALL  HTRIBK(NM,N,AR,AI,FM1,N,ZR,ZI)
   50 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F297-2 CINVIT

A Fortran IV Subroutine to Determine Those Eigenvectors of a Complex Upper Hessenberg Matrix Corresponding to Specified Eigenvalues.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine CINVIT computes the eigenvectors of a complex upper Hessenberg matrix corresponding to specified eigenvalues using inverse iteration.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  CINVIT(NM,N,AR,AI,WR,WI,SELECT,
                   MM,M,ZR,ZI,IERR,RM1,RM2,RV1,RV2)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays AR, AI, ZR, and ZI as specified in the DIMENSION statements for AR, AI, ZR, and ZI in the calling program.

N — is an integer input variable set equal to the order of the matrix A = (AR,AI). N must be not greater than NM.

AR,AI — are working precision real input two-dimensional variables with row dimension NM and column dimension at least N containing the real and imaginary parts, respectively, of the complex upper Hessenberg matrix.

WR,WI are working precision real one-dimensional variables of dimension at least N. On input, they contain the real and imaginary parts, respectively, of the eigenvalues of the Hessenberg matrix. The eigenvalues need not be ordered, except that eigenvalues of any submatrix of the input Hessenberg matrix must have indices in WR,WI which lie between the boundary indices for that submatrix. These ordering constraints are satisfied by the output eigenvalues from COMLR (F295). On output, the eigenvalues are unaltered, except that the real parts of close eigenvalues may be perturbed slightly in an attempt to obtain independent eigenvectors.

SELECT is a logical input one-dimensional variable of dimension at least N whose true elements flag those eigenvalues whose eigenvectors are desired.

MM is an integer input variable set equal to an upper bound for the number of columns required to hold the desired eigenvectors.

M is an integer output variable set equal to the number of columns actually used to store the eigenvectors.

ZR,ZI are working precision real output two-dimensional variables with row dimension NM and column dimension at least MM containing the real and imaginary parts, respectively, of the eigenvectors corresponding to the flagged eigenvalues. The eigenvectors are packed into the columns of ZR and ZI starting at the first columns.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

RM1,RM2 are working precision real temporary array variables of dimension at least N*N. These arrays holds the triangularized form of the upper Hessenberg matrix used in the inverse iteration process.

RV1,RV2 are working precision real temporary one-dimensional variables of dimension at least N. They hold the approximate eigenvectors during the inverse iteration process.

B. Error Conditions and Returns.

If more than MM eigenvectors are requested, CINVIT terminates with IERR set to -(2*N+1) and with the first MM eigenvectors in the arrays ZR and ZI. In this case, M = MM.

If none of the initial vectors for the inverse iteration process produces an acceptable approximation to an eigenvector, CINVIT terminates the computation for that eigenvector and sets IERR to -K where K is the index of the associated eigenvalue. If this failure occurs for more than one eigenvalue, the last occurrence is recorded in IERR. The columns of ZR and ZI corresponding to failures of the above sort are set to zero vectors.

If both of the above error conditions occur, CINVIT terminates with IERR set to -(N+K) where K is the index of the last eigenvalue whose inverse iteration process failed. The columns of ZR and ZI corresponding to failures in the inverse iteration process are set to zero vectors, and in this case, M = MM.

If neither of the above error conditions occurs, CINVIT sets IERR to zero.

C. Applicability and Restrictions.

To determine eigenvectors corresponding to some of the eigenvalues of a complex general matrix, CINVIT should be preceded by COMHES (F282) and COMLR (F295). COMHES provides a suitable complex upper Hessenberg matrix for CINVIT and COMLR determines the eigenvalues of this Hessenberg matrix. CINVIT should then be followed by COMBAK (F283) to back transform the eigenvectors from CINVIT into those of the original matrix. Note: the upper Hessenberg matrix must be saved before COMLR, since COMLR destroys this part of A = (AR,AI).

It is recommended in general that CBAL (F271) be used before COMHES in which case CBABK2 (F272) must be used after COMBAK.

In this implementation, the arithmetic is real throughout except for complex division and complex absolute value.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvectors are determined by inverse iteration. In CINVIT, this is a process whereby a vector X1 satisfying the matrix linear equation U*X1 = X0 is computed, where X0 is an initial vector and U is the upper triangular factor in the LU decomposition of the matrix B-W*I with partial pivoting. W is an approximate eigenvalue of a leading submatrix B of the Hessenberg matrix in A. The vector X1 is accepted as an eigenvector of B if the norm of X1 is sufficiently larger than the norm of X0. This eigenvector of B is transformed to an eigenvector of A by simply appending zero components to it.

The real and imaginary parts of U are stored in the two real arrays RM1 and RM2 respectively. Internally, RM1 and RM2 are treated as two-dimensional NxN arrays.

The acceptance criterion for X1 is a growth test of its norm. If MACHEP denotes the relative machine precision, then currently, X1 is rejected as an eigenvector if

SQRT(UK)*!!B!!*MACHEP*!!X1!! .LT. !!X0!!/10.0

where UK is the order of the submatrix B whose eigenvector is being computed and the norm !! !! is the infinity norm. At most UK orthogonal initial vectors X0 are tried to obtain the required growth. If no vector is accepted, the parameter IERR is set to indicate this failure and CINVIT proceeds to compute the next eigenvector.

The real parts of close or identical eigenvalues whose eigenvectors are desired may be perturbed slightly in an attempt to obtain independent eigenvectors (if they exist). If required, such perturbations are positive and small multiples of MACHEP*!!B!!.

This subroutine is a translation of the Algol procedure CX INVIT written and discussed in detail by Peters and Wilkinson (1).

4. REFERENCES.

1) Peters, G. and Wilkinson, J.H., The Calculation of Specified Eigenvectors by Inverse Iteration, Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/18, 418-439, Springer-Verlag, 1971.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The accuracy of CINVIT can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of complex upper Hessenberg matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original upper Hessenberg matrix.

```
      SUBROUTINE CINVIT(NM,N,AR,AI,WR,WI,SELECT,MM,M,ZR,ZI,
     X                  IERR,RM1,RM2,RV1,RV2)
C
      INTEGER I,J,K,M,N,S,II,MM,MP,NM,UK,IP1,ITS,KM1,IERR
      REAL AR(NM,N),AI(NM,N),WR(N),WI(N),ZR(NM,MM),ZI(NM,MM),
     X       RM1(N,N),RM2(N,N),RV1(N),RV2(N)
      REAL X,Y,EPS3,NORM,NORMV,GROWTO,ILAMBD,MACHEP,RLAMBD,UKROOT
      REAL SQRT,CABS,ABS,FLOAT
      LOGICAL SELECT(N)
      COMPLEX Z3
      COMPLEX CMPLX
      REAL REAL,AIMAG
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      UK = 0
      S = 1
C
      DO 980 K = 1, N
         IF (.NOT. SELECT(K)) GO TO 980
         IF (S .GT. MM) GO TO 1000
         IF (UK .GE. K) GO TO 200
C     ********** CHECK FOR POSSIBLE SPLITTING **********
         DO 120 UK = K, N
            IF (UK .EQ. N) GO TO 140
            IF (AR(UK+1,UK) .EQ. 0.0 .AND. AI(UK+1,UK) .EQ. 0.0)
     X         GO TO 140
  120    CONTINUE
C     ********** COMPUTE INFINITY NORM OF LEADING UK BY UK
C                (HESSENBERG) MATRIX **********
  140    NORM = 0.0
         MP = 1
C
         DO 180 I = 1, UK
            X = 0.0
C
            DO 160 J = MP, UK
  160       X = X + CABS(CMPLX(AR(I,J),AI(I,J)))
C
            IF (X .GT. NORM) NORM = X
            MP = I
  180    CONTINUE
C     ********** EPS3 REPLACES ZERO PIVOT IN DECOMPOSITION
C                AND CLOSE ROOTS ARE MODIFIED BY EPS3 **********
         IF (NORM .EQ. 0.0) NORM = 1.0
         EPS3 = MACHEP * NORM
```

```
C      ********** GROWTO IS THE CRITERION FOR GROWTH **********
          UKROOT = SQRT(FLOAT(UK))
          GROWTO = 1.0E-1 / UKROOT
  200     RLAMBD = WR(K)
          ILAMBD = WI(K)
          IF (K .EQ. 1) GO TO 280
          KM1 = K - 1
          GO TO 240
C      ********** PERTURB EIGENVALUE IF IT IS CLOSE
C                 TO ANY PREVIOUS EIGENVALUE **********
  220     RLAMBD = RLAMBD + EPS3
C      ********** FOR I=K-1 STEP -1 UNTIL 1 DO -- **********
  240     DO 260 II = 1, KM1
             I = K - II
             IF (SELECT(I) .AND. ABS(WR(I)-RLAMBD) .LT. EPS3 .AND.
     X          ABS(WI(I)-ILAMBD) .LT. EPS3) GO TO 220
  260     CONTINUE
C
          WR(K) = RLAMBD
C      ********** FORM UPPER HESSENBERG (AR,AI)-(RLAMBD,ILAMBD)*I
C                 AND INITIAL COMPLEX VECTOR **********
  280     MP = 1
C
          DO 320 I = 1, UK
C
             DO 300 J = MP, UK
                RM1(I,J) = AR(I,J)
                RM2(I,J) = AI(I,J)
  300        CONTINUE
C
             RM1(I,I) = RM1(I,I) - RLAMBD
             RM2(I,I) = RM2(I,I) - ILAMBD
             MP = I
             RV1(I) = EPS3
  320     CONTINUE
C      ********** TRIANGULAR DECOMPOSITION WITH INTERCHANGES,
C                 REPLACING ZERO PIVOTS BY EPS3 **********
          IF (UK .EQ. 1) GO TO 420
C
          DO 400 I = 2, UK
             MP = I - 1
             IF (CABS(CMPLX(RM1(I,MP),RM2(I,MP))) .LE.
     X           CABS(CMPLX(RM1(MP,MP),RM2(MP,MP)))) GO TO 360
C
             DO 340 J = MP, UK
                Y = RM1(I,J)
                RM1(I,J) = RM1(MP,J)
                RM1(MP,J) = Y
                Y = RM2(I,J)
                RM2(I,J) = RM2(MP,J)
                RM2(MP,J) = Y
  340        CONTINUE
```

```
C
  360        IF (RM1(MP,MP) .EQ. 0.0 .AND. RM2(MP,MP) .EQ. 0.0)
     X          RM1(MP,MP) = EPS3
             Z3 = CMPLX(RM1(I,MP),RM2(I,MP)) /
     X               CMPLX(RM1(MP,MP),RM2(MP,MP))
             X = REAL(Z3)
             Y = AIMAG(Z3)
             IF (X .EQ. 0.0 .AND. Y .EQ. 0.0) GO TO 400
C
             DO 380 J = I, UK
                RM1(I,J) = RM1(I,J) - X * RM1(MP,J) + Y * RM2(MP,J)
                RM2(I,J) = RM2(I,J) - X * RM2(MP,J) - Y * RM1(MP,J)
  380        CONTINUE
C
  400     CONTINUE
C
  420     IF (RM1(UK,UK) .EQ. 0.0 .AND. RM2(UK,UK) .EQ. 0.0)
     X       RM1(UK,UK) = EPS3
          ITS = 0
C     ********** BACK SUBSTITUTION
C                  FOR I=UK STEP -1 UNTIL 1 DO -- **********
  660     DO 720 II = 1, UK
             I = UK + 1 - II
             X = RV1(I)
             Y = 0.0
             IF (I .EQ. UK) GO TO 700
             IP1 = I + 1
C
             DO 680 J = IP1, UK
                X = X - RM1(I,J) * RV1(J) + RM2(I,J) * RV2(J)
                Y = Y - RM1(I,J) * RV2(J) - RM2(I,J) * RV1(J)
  680        CONTINUE
C
  700        Z3 = CMPLX(X,Y) / CMPLX(RM1(I,I),RM2(I,I))
             RV1(I) = REAL(Z3)
             RV2(I) = AIMAG(Z3)
  720     CONTINUE
C     ********** ACCEPTANCE TEST FOR EIGENVECTOR
C                  AND NORMALIZATION **********
          ITS = ITS + 1
          NORM = 0.0
          NORMV = 0.0
C
          DO 780 I = 1, UK
             X = CABS(CMPLX(RV1(I),RV2(I)))
             IF (NORMV .GE. X) GO TO 760
             NORMV = X
             J = I
  760        NORM = NORM + X
  780     CONTINUE
C
          IF (NORM .LT. GROWTO) GO TO 840
```

```
C     ********** ACCEPT VECTOR **********
         X = RV1(J)
         Y = RV2(J)
C
         DO 820 I = 1, UK
            Z3 = CMPLX(RV1(I),RV2(I)) / CMPLX(X,Y)
            ZR(I,S) = REAL(Z3)
            ZI(I,S) = AIMAG(Z3)
  820    CONTINUE
C
         IF (UK .EQ. N) GO TO 940
         J = UK + 1
         GO TO 900
C     ********** IN-LINE PROCEDURE FOR CHOOSING
C                A NEW STARTING VECTOR **********
  840    IF (ITS .GE. UK) GO TO 880
         X = UKROOT
         Y = EPS3 / (X + 1.0)
         RV1(1) = EPS3
C
         DO 860 I = 2, UK
  860    RV1(I) = Y
C
         J = UK - ITS + 1
         RV1(J) = RV1(J) - EPS3 * X
         GO TO 660
C     ********** SET ERROR -- UNACCEPTED EIGENVECTOR **********
  880    J = 1
         IERR = -K
C     ********** SET REMAINING VECTOR COMPONENTS TO ZERO **********
  900    DO 920 I = J, N
            ZR(I,S) = 0.0
            ZI(I,S) = 0.0
  920    CONTINUE
C
  940    S = S + 1
  980 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- UNDERESTIMATE OF EIGENVECTOR
C                SPACE REQUIRED **********
 1000 IF (IERR .NE. 0) IERR = IERR - N
      IF (IERR .EQ. 0) IERR = -(2 * N + 1)
 1001 M = S - 1
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F283-2 COMBAK

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Upper Hessenberg Matrix Determined by COMHES.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine COMBAK forms the eigenvectors of a complex general matrix from the eigenvectors of that upper Hessenberg matrix determined by COMHES (F282).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE COMBAK(NM,LOW,IGH,AR,AI,INT,M,ZR,ZI)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays AR, AI, ZR, and ZI as specified in the DIMENSION statements for AR, AI, ZR, and ZI in the calling program.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F271 for the details. If the matrix is not balanced, set LOW to 1 and IGH to the order of the matrix.

AR,AI are working precision real input two-dimensional variables with row dimension NM and column dimension at least IGH. Their lower triangles below the subdiagonal contain the multipliers which were used in

the reduction to the Hessenberg form. The remaining upper parts of AR and AI are arbitrary. See section 3 of F282 for the details.

INT is an integer input one-dimensional variable of dimension at least IGH. INT identifies the rows and columns interchanged during the reduction by COMHES. See section 3 of F282 for the details.

M is an integer input variable set equal to the number of columns of Z = (ZR,ZI) to be back transformed.

ZR,ZI are working precision real two-dimensional variables with row dimension NM and column dimension at least M. On input, the first M columns of ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors to be back transformed. On output, these M columns of ZR and ZI contain the real and imaginary parts of the transformed eigenvectors.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine COMHES (F282).

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix C (say) has been reduced to the upper Hessenberg form F stored in A by the similarity transformation

$$F = G^{-1} C G$$

where G is a product of the permutation and elementary matrices encoded in INT and in a lower triangle of A under F respectively. Then, given an array Z of column vectors, COMBAK computes the matrix product GZ. If the eigenvectors of F are columns of the array Z, then COMBAK forms the eigenvectors of C in their place.

This subroutine is a translation of the Algol procedure COMBAK written and discussed in detail by Martin and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., Similarity Reduction of a General Matrix to Hessenberg Form, Num. Math. 12,349-368 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/13, 339-358, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The accuracy of COMBAK can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of complex general matrices. In these paths, this subroutine is, in practice, numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix usually close to the original matrix.

```
      SUBROUTINE COMBAK(NM,LOW,IGH,AR,AI,INT,M,ZR,ZI)
C
      INTEGER I,J,M,LA,MM,MP,NM,IGH,KP1,LOW,MP1
      REAL AR(NM,IGH),AI(NM,IGH),ZR(NM,M),ZI(NM,M)
      REAL XR,XI
      INTEGER INT(IGH)
C
      IF (M .EQ. 0) GO TO 200
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C     ********** FOR MP=IGH-1 STEP -1 UNTIL LOW+1 DO -- **********
      DO 140 MM = KP1, LA
         MP = LOW + IGH - MM
         MP1 = MP + 1
C
         DO 110 I = MP1, IGH
            XR = AR(I,MP-1)
            XI = AI(I,MP-1)
            IF (XR .EQ. 0.0 .AND. XI .EQ. 0.0) GO TO 110
C
            DO 100 J = 1, M
               ZR(I,J) = ZR(I,J) + XR * ZR(MP,J) - XI * ZI(MP,J)
               ZI(I,J) = ZI(I,J) + XR * ZI(MP,J) + XI * ZR(MP,J)
  100       CONTINUE
C
  110    CONTINUE
C
         I = INT(MP)
         IF (I .EQ. MP) GO TO 140
C
         DO 130 J = 1, M
            XR = ZR(I,J)
            ZR(I,J) = ZR(MP,J)
            ZR(MP,J) = XR
            XI = ZI(I,J)
            ZI(I,J) = ZI(MP,J)
            ZI(MP,J) = XI
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F282-2 COMHES

A Fortran IV Subroutine to Reduce a Complex General Matrix to Complex Upper Hessenberg Form Using Elementary Transformations.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine COMHES reduces a complex general matrix to complex upper Hessenberg form using stabilized elementary similarity transformations. This reduced form is used by other subroutines to find the eigenvalues and/or eigenvectors of the original matrix. See section 2C for the specific routines.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  COMHES(NM,N,LOW,IGH,AR,AI,INT)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays AR and AI as specified in the DIMENSION statements for AR and AI in the calling program.

N is an integer input variable set equal to the order of the matrix A = (AR,AI). N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F271 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

AR,AI are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, AR and AI contain the real and imaginary parts, respectively, of the complex matrix of order N to be reduced to Hessenberg form. On output, AR and AI contain the real and imaginary parts of the upper Hessenberg matrix as well as the multipliers used in the reduction. See section 3 for the details.

INT is an integer output one-dimensional variable of dimension at least IGH identifying the rows and columns interchanged during the reduction. See section 3 for the details. Only components LOW+1 through IGH-1 are actually used by COMHES.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues of the original matrix are desired, this subroutine should be followed by COMLR (F295).

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by COMLR2 (F296).

If some of the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by COMLR (F295), CINVIT (F297), and COMBAK (F283).

In this implementation, the arithmetic is real throughout except for complex division.

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix A has the form

$$A = \begin{pmatrix} T & X & Y \\ 0 & B & Z \\ 0 & 0 & R \end{pmatrix}$$

where T and R are upper triangular matrices, B is a square matrix situated in rows and columns LOW through IGH, and X, Y, and Z are rectangular matrices of the appropriate dimensions. Then the subroutine COMHES performs permutation and elementary similarity transformations to reduce B to Hessenberg form.

The Hessenberg reduction is performed in the following way. Starting with J=LOW, a permutation similarity transformation is performed to stabilize the succeeding elementary transformations in the J-th column of A. Next, each non-zero element below the subdiagonal in the J-th column of A is eliminated using elementary similarity transformations.

The permutation similarity transformation is determined by searching for the element of maximum modulus in the J-th column below the subdiagonal. This element is placed into the subdiagonal position by a row interchange and the corresponding column interchange is made to complete the permutation transformation. The index of the row interchanged with the (J+1)-th row is stored in INT(J+1).

The elementary transformation consists of elementary row operations on the J-th column of A to eliminate the non-zero elements below the diagonal and the corresponding elementary column operations to complete the similarity. The multipliers used in this elimination process are stored in place of the eliminated elements. These multipliers as well as the row interchanges stored in INT are used later in COMBAK for the back transformation of the eigenvectors.

The above steps are repeated on further columns of the transformed A until B is reduced to Hessenberg form; that is, repeated for J = LOW+1,LOW+2,...,IGH-2.

This subroutine is a translation of the Algol procedure COMHES written and discussed in detail by Martin and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., Similarity Reduction of a General Matrix to Hessenberg Form, Num. Math. 12,349-368 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/13, 339-358, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The accuracy of COMHES can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of complex general matrices. In these paths, this subroutine is, in practice, numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix usually close to the original matrix.

```
      SUBROUTINE COMHES(NM,N,LOW,IGH,AR,AI,INT)
C
      INTEGER I,J,M,N,LA,NM,IGH,KP1,LOW,MM1,MP1
      REAL AR(NM,N),AI(NM,N)
      REAL XR,XI,YR,YI
      REAL ABS
      INTEGER INT(IGH)
      COMPLEX Z3
      COMPLEX CMPLX
      REAL REAL,AIMAG
C
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C
      DO 180 M = KP1, LA
         MM1 = M - 1
         XR = 0.0
         XI = 0.0
         I = M
C
         DO 100 J = M, IGH
            IF (ABS(AR(J,MM1)) + ABS(AI(J,MM1))
     X         .LE. ABS(XR) + ABS(XI)) GO TO 100
            XR = AR(J,MM1)
            XI = AI(J,MM1)
            I = J
  100    CONTINUE
C
         INT(M) = I
         IF (I .EQ. M) GO TO 130
C     ********** INTERCHANGE ROWS AND COLUMNS OF AR AND AI **********
         DO 110 J = MM1, N
            YR = AR(I,J)
            AR(I,J) = AR(M,J)
            AR(M,J) = YR
            YI = AI(I,J)
            AI(I,J) = AI(M,J)
            AI(M,J) = YI
  110    CONTINUE
C
         DO 120 J = 1, IGH
            YR = AR(J,I)
            AR(J,I) = AR(J,M)
            AR(J,M) = YR
            YI = AI(J,I)
            AI(J,I) = AI(J,M)
            AI(J,M) = YI
  120    CONTINUE
C     ********** END INTERCHANGE **********
  130    IF (XR .EQ. 0.0 .AND. XI .EQ. 0.0) GO TO 180
         MP1 = M + 1
```

```
C
      DO 160 I = MP1, IGH
         YR = AR(I,MM1)
         YI = AI(I,MM1)
         IF (YR .EQ. 0.0 .AND. YI .EQ. 0.0) GO TO 160
         Z3 = CMPLX(YR,YI) / CMPLX(XR,XI)
         YR = REAL(Z3)
         YI = AIMAG(Z3)
         AR(I,MM1) = YR
         AI(I,MM1) = YI
C
         DO 140 J = M, N
            AR(I,J) = AR(I,J) - YR * AR(M,J) + YI * AI(M,J)
            AI(I,J) = AI(I,J) - YR * AI(M,J) - YI * AR(M,J)
  140    CONTINUE
C
         DO 150 J = 1, IGH
            AR(J,M) = AR(J,M) + YR * AR(J,I) - YI * AI(J,I)
            AI(J,M) = AI(J,M) + YR * AI(J,I) + YI * AR(J,I)
  150    CONTINUE
C
  160 CONTINUE
C
  180 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F295-2 COMLR

A Fortran IV Subroutine to Determine the Eigenvalues of a Complex Upper Hessenberg Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine COMLR computes the eigenvalues of a complex upper Hessenberg matrix using the modified LR method.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  COMLR(NM,N,LOW,IGH,HR,HI,WR,WI,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays HR and HI as specified in the DIMENSION statements for HR and HI in the calling program.

N is an integer input variable set equal to the order of the matrix H = (HR,HI). N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F271 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

HR,HI are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, HR and HI contain the real and imaginary parts, respectively, of the complex upper Hessenberg matrix. Note: COMLR destroys this upper Hessenberg matrix.

WR,WI are working precision real output one-dimensional variables of dimension at least N containing the real and imaginary parts, respectively, of the eigenvalues of the upper Hessenberg matrix.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, this subroutine terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the WR and WI arrays should be correct for indices IERR+1,IERR+2,...,N.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine some of the eigenvectors of a complex Hessenberg matrix, COMLR should be followed by CINVIT (F297) to compute those eigenvectors. Note, however, that the upper Hessenberg matrix must be saved before COMLR for later use of CINVIT, since COMLR destroys this upper Hessenberg matrix.

To determine the eigenvalues of a complex general matrix, COMLR should be preceded by COMHES (F282) to provide a suitable complex upper Hessenberg matrix for COMLR.

To determine some of the eigenvectors of a complex general matrix, COMLR should be preceded by COMHES (F282) and followed by CINVIT (F297) and COMBAK (F283). COMBAK back transforms the eigenvectors from CINVIT into those of the complex general matrix. Note, as above, that the Hessenberg matrix must be saved before COMLR.

It is recommended in general that CBAL (F271) be used before COMHES in which case CBABK2 (F272) must be used after COMBAK if eigenvectors are computed.

The subroutine COMLR executes faster than its counterpart COMQR (F245) which uses unitary similarity transformations. COMQR is, however, more accurate in some cases. It is recommended that COMQR be used in general.

In this implementation, the arithmetic is real throughout except for complex square root and complex division.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the modified LR method. The essence of this method is a process whereby a sequence of complex upper Hessenberg matrices, similar to the original Hessenberg matrix, is formed which converges to a triangular matrix. The factorization of H into upper (R) and lower (L) triangular matrices is stabilized by permutation similarity transformations at each stage. The rate of convergence of this sequence is improved by shifting the origin at each iteration. Before each iteration, the last Hessenberg form is checked for a possible splitting into submatrices. If a splitting occurs, only the lower submatrix participates in the next iteration.

The origin shift at each iteration is the eigenvalue of the lowest 2x2 principal minor closer to the second diagonal element of this minor. Whenever a lowest 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision.

Some of the eigenvalues may have been isolated on the diagonal by the subroutine CBAL (F271). This information is transmitted to COMLR through the parameters LOW and IGH. As a result, COMLR immediately extracts the eigenvalues in rows 1 to LOW-1 and IGH+1 to N, and so applies the LR procedure to the submatrix situated in rows and columns LOW through IGH.

This subroutine is a translation of the Algol procedure COMLR written and discussed in detail by Martin and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., The Modified LR Algorithm for Complex Hessenberg Matrices, Num. Math. 12,369-376 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/16, 396-403, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The subroutine COMLR is usually numerically stable (1); that is, each computed eigenvalue is exact for a matrix usually close to the original upper Hessenberg matrix.

```
      SUBROUTINE COMLR(NM,N,LOW,IGH,HR,HI,WR,WI,IERR)
C
      INTEGER I,J,L,M,N,EN,LL,MM,NM,IGH,IM1,ITS,LOW,MP1,ENM1,IERR
      REAL HR(NM,N),HI(NM,N),WR(N),WI(N)
      REAL SI,SR,TI,TR,XI,XR,YI,YR,ZZI,ZZR,MACHEP
      REAL ABS
      COMPLEX Z3
      COMPLEX CSQRT,CMPLX
      REAL REAL,AIMAG
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
C     ********** STORE ROOTS ISOLATED BY CBAL **********
      DO 200 I = 1, N
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 200
         WR(I) = HR(I,I)
         WI(I) = HI(I,I)
  200 CONTINUE
C
      EN = IGH
      TR = 0.0
      TI = 0.0
C     ********** SEARCH FOR NEXT EIGENVALUE **********
  220 IF (EN .LT. LOW) GO TO 1001
      ITS = 0
      ENM1 = EN - 1
C     ********** LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW  -- **********
  240 DO 260 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 300
         IF (ABS(HR(L,L-1)) + ABS(HI(L,L-1)) .LE.
     X       MACHEP * (ABS(HR(L-1,L-1)) + ABS(HI(L-1,L-1))
     X             + ABS(HR(L,L)) + ABS(HI(L,L)))) GO TO 300
  260 CONTINUE
C     ********** FORM SHIFT **********
  300 IF (L .EQ. EN) GO TO 660
      IF (ITS .EQ. 30) GO TO 1000
      IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
      SR = HR(EN,EN)
      SI = HI(EN,EN)
      XR = HR(ENM1,EN) * HR(EN,ENM1) - HI(ENM1,EN) * HI(EN,ENM1)
      XI = HR(ENM1,EN) * HI(EN,ENM1) + HI(ENM1,EN) * HR(EN,ENM1)
      IF (XR .EQ. 0.0 .AND. XI .EQ. 0.0) GO TO 340
      YR = (HR(ENM1,ENM1) - SR) / 2.0
      YI = (HI(ENM1,ENM1) - SI) / 2.0
      Z3 = CSQRT(CMPLX(YR**2-YI**2+XR,2.0*YR*YI+XI))
      ZZR = REAL(Z3)
      ZZI = AIMAG(Z3)
```

```
      IF (YR * ZZR + YI * ZZI .GE. 0.0) GO TO 310
      ZZR = -ZZR
      ZZI = -ZZI
  310 Z3 = CMPLX(XR,XI) / CMPLX(YR+ZZR,YI+ZZI)
      SR = SR - REAL(Z3)
      SI = SI - AIMAG(Z3)
      GO TO 340
C     ********** FORM EXCEPTIONAL SHIFT **********
  320 SR = ABS(HR(EN,ENM1)) + ABS(HR(ENM1,EN-2))
      SI = ABS(HI(EN,ENM1)) + ABS(HI(ENM1,EN-2))
C
  340 DO 360 I = LOW, EN
         HR(I,I) = HR(I,I) - SR
         HI(I,I) = HI(I,I) - SI
  360 CONTINUE
C
      TR = TR + SR
      TI = TI + SI
      ITS = ITS + 1
C     ********** LOOK FOR TWO CONSECUTIVE SMALL
C                SUB-DIAGONAL ELEMENTS **********
      XR = ABS(HR(ENM1,ENM1)) + ABS(HI(ENM1,ENM1))
      YR = ABS(HR(EN,ENM1)) + ABS(HI(EN,ENM1))
      ZZR = ABS(HR(EN,EN)) + ABS(HI(EN,EN))
C     ********** FOR M=EN-1 STEP -1 UNTIL L DO -- **********
      DO 380 MM = L, ENM1
         M = ENM1 + L - MM
         IF (M .EQ. L) GO TO 420
         YI = YR
         YR = ABS(HR(M,M-1)) + ABS(HI(M,M-1))
         XI = ZZR
         ZZR = XR
         XR = ABS(HR(M-1,M-1)) + ABS(HI(M-1,M-1))
         IF (YR .LE. MACHEP * ZZR / YI * (ZZR + XR + XI)) GO TO 420
  380 CONTINUE
C     ********** TRIANGULAR DECOMPOSITION H=L*R **********
  420 MP1 = M + 1
C
      DO 520 I = MP1, EN
         IM1 = I - 1
         XR = HR(IM1,IM1)
         XI = HI(IM1,IM1)
         YR = HR(I,IM1)
         YI = HI(I,IM1)
         IF (ABS(XR) + ABS(XI) .GE. ABS(YR) + ABS(YI)) GO TO 460
C     ********** INTERCHANGE ROWS OF HR AND HI **********
         DO 440 J = IM1, EN
            ZZR = HR(IM1,J)
            HR(IM1,J) = HR(I,J)
            HR(I,J) = ZZR
            ZZI = HI(IM1,J)
            HI(IM1,J) = HI(I,J)
            HI(I,J) = ZZI
  440    CONTINUE
```

```
C
         Z3 = CMPLX(XR,XI) / CMPLX(YR,YI)
         WR(I) = 1.0
         GO TO 480
  460    Z3 = CMPLX(YR,YI) / CMPLX(XR,XI)
         WR(I) = -1.0
  480    ZZR = REAL(Z3)
         ZZI = AIMAG(Z3)
         HR(I,IM1) = ZZR
         HI(I,IM1) = ZZI
C
         DO 500 J = I, EN
            HR(I,J) = HR(I,J) - ZZR * HR(IM1,J) + ZZI * HI(IM1,J)
            HI(I,J) = HI(I,J) - ZZR * HI(IM1,J) - ZZI * HR(IM1,J)
  500    CONTINUE
C
  520 CONTINUE
C     ********** COMPOSITION R*L=H **********
      DO 640 J = MP1, EN
         XR = HR(J,J-1)
         XI = HI(J,J-1)
         HR(J,J-1) = 0.0
         HI(J,J-1) = 0.0
C     ********** INTERCHANGE COLUMNS OF HR AND HI,
C                IF NECESSARY **********
         IF (WR(J) .LE. 0.0) GO TO 580
C
         DO 540 I = L, J
            ZZR = HR(I,J-1)
            HR(I,J-1) = HR(I,J)
            HR(I,J) = ZZR
            ZZI = HI(I,J-1)
            HI(I,J-1) = HI(I,J)
            HI(I,J) = ZZI
  540    CONTINUE
C
  580    DO 600 I = L, J
            HR(I,J-1) = HR(I,J-1) + XR * HR(I,J) - XI * HI(I,J)
            HI(I,J-1) = HI(I,J-1) + XR * HI(I,J) + XI * HR(I,J)
  600    CONTINUE
C
  640 CONTINUE
C
      GO TO 240
C     ********** A ROOT FOUND **********
  660 WR(EN) = HR(EN,EN) + TR
      WI(EN) = HI(EN,EN) + TI
      EN = ENM1
      GO TO 220
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = EN
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F296-2 COMLR2

A Fortran IV Subroutine to Determine the Eigenvalues and Eigenvectors of a Complex Upper Hessenberg Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine COMLR2 computes the eigenvalues and eigenvectors of a complex upper Hessenberg matrix. COMLR2 uses the modified LR method to compute the eigenvalues and accumulates the LR transformations to compute the eigenvectors. The eigenvectors of a complex general matrix can also be computed directly by COMLR2, if COMHES (F282) has been used to reduce this matrix to Hessenberg form.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  COMLR2(NM,N,LOW,IGH,INT,HR,HI,
                   WR,WI,ZR,ZI,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays HR, HI, ZR, and ZI as specified in the DIMENSION statements for HR, HI, ZR, and ZI in the calling program.

N — is an integer input variable set equal to the order of the matrix H = (HR,HI). N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F271 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

INT is an integer input one-dimensional variable of dimension at least IGH. Only components LOW through IGH are used by COMLR2. If the eigenvectors of the Hessenberg matrix are desired, set INT(J)=J for J = LOW,LOW+1,...,IGH. If the eigenvectors of a complex general matrix are desired, INT contains information saved by COMHES identifying the rows and columns interchanged during the reduction to Hessenberg form.

HR,HI are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, HR and HI contain the real and imaginary parts, respectively, of the complex upper Hessenberg matrix. If the eigenvectors of the Hessenberg matrix are desired, set the lower triangles of HR,HI below the subdiagonal to zero. If the eigenvectors of a complex general matrix are desired, the lower triangles of HR,HI contain the multipliers which are generated by COMHES in the reduction to the Hessenberg form. Note: COMLR2 destroys the upper Hessenberg portions of HR,HI, but returns in location HR(1,1) the norm of the triangularized matrix. See section 2C for the definition and significance of this quantity.

WR,WI are working precision real output one-dimensional variables of dimension at least N containing the real and imaginary parts, respectively, of the eigenvalues of the upper Hessenberg matrix.

ZR,ZI are working precision real output two-dimensional variables with row dimension NM and column dimension at least N containing the real and imaginary parts, respectively, of the eigenvectors. The eigenvectors are not normalized.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, this subroutine terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the WR and WI arrays should be correct for indices IERR+1,IERR+2,...,N, but no eigenvectors are computed.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine the eigenvalues and eigenvectors of a complex general matrix, COMLR2 must be preceded by COMHES (F282) to provide a suitable complex Hessenberg matrix for COMLR2.

It is recommended in general that CBAL (F271) be used before COMHES in which case CBABK2 (F272) must be used after COMLR2.

The subroutine COMLR2 may occasionally return poor results, especially in the eigenvectors, due to pronounced growth in the matrix elements that may occur during the LR iterations. A measure of this growth can be obtained from a comparison of the norm quantity returned in HR(1,1), defined as the sum of the absolute values of the real and imaginary components of all the elements of the triangularized matrix, with the corresponding norm of the vector of eigenvalues. This growth cannot occur in the QR algorithm embodied in subroutine COMQR2 (F246), the unitary counterpart of COMLR2. It is recommended that COMQR2, although slower, be used in general.

In this implementation, the arithmetic is real throughout except for complex square root and complex division.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the modified LR method. The essence of this method is a process whereby a sequence of complex upper Hessenberg matrices, similar to the original Hessenberg matrix, is formed which converges to a triangular matrix. The factorization of H into upper (R) and lower (L) triangular matrices is stabilized by permutation similarity transformations at each stage. The rate of convergence of this sequence is improved by shifting the origin at each iteration. Before each iteration, the last Hessenberg form is checked for a possible splitting into submatrices. If a splitting occurs, only the lower submatrix participates in the next iteration. The similarity transformations used in each iteration are accumulated in the ZR and ZI arrays along with the reduction transformations for the original matrix.

The origin shift at each iteration is the eigenvalue of the lowest 2x2 principal minor closer to the second diagonal element of this minor. Whenever a lowest 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision. The eigenvectors of this triangular matrix are determined by back substitution, and then back transformed into those of the original matrix.

Some of the eigenvalues may have been isolated on the diagonal by the subroutine CBAL (F271). This information is transmitted to COMLR2 through the parameters LOW and IGH. As a result, COMLR2 immediately extracts the eigenvalues in rows 1 to LOW-1 and IGH+1 to N, and so applies the LR procedure to the submatrix situated in rows and columns LOW through IGH.

This subroutine is a translation of the Algol procedure COMLR2 written and discussed in detail by Peters and Wilkinson (1).

4. REFERENCES.

1) Peters, G. and Wilkinson, J.H., Eigenvectors of Real and Complex Matrices by LR and QR Triangularizations, Num. Math. 16,181-204 (1970). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/15, 372-395, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The subroutine COMLR2 is usually numerically stable (1); that is, each computed eigenvalue and its corresponding eigenvector are exact for a matrix usually close to the original upper Hessenberg matrix.

```
      SUBROUTINE COMLR2(NM,N,LOW,IGH,INT,HR,HI,WR,WI,ZR,ZI,IERR)
C
      INTEGER I,J,K,L,M,N,EN,II,JJ,LL,MM,NM,NN,IGH,IM1,IP1,
     X        ITS,LOW,MP1,ENM1,IEND,IERR
      REAL HR(NM,N),HI(NM,N),WR(N),WI(N),ZR(NM,N),ZI(NM,N)
      REAL SI,SR,TI,TR,XI,XR,YI,YR,ZZI,ZZR,NORM,MACHEP
      REAL ABS
      INTEGER INT(IGH)
      INTEGER MIN0
      COMPLEX Z3
      COMPLEX CSQRT,CMPLX
      REAL REAL,AIMAG
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
C     ********** INITIALIZE EIGENVECTOR MATRIX **********
      DO 100 I = 1, N
C
         DO 100 J = 1, N
            ZR(I,J) = 0.0
            ZI(I,J) = 0.0
            IF (I .EQ. J) ZR(I,J) = 1.0
  100 CONTINUE
C     ********** FORM THE MATRIX OF ACCUMULATED TRANSFORMATIONS
C                FROM THE INFORMATION LEFT BY COMHES **********
      IEND = IGH - LOW - 1
      IF (IEND .LE. 0) GO TO 180
C     ********** FOR I=IGH-1 STEP -1 UNTIL LOW+1 DO -- **********
      DO 160 II = 1, IEND
         I = IGH - II
         IP1 = I + 1
C
         DO 120 K = IP1, IGH
            ZR(K,I) = HR(K,I-1)
            ZI(K,I) = HI(K,I-1)
  120    CONTINUE
C
         J = INT(I)
         IF (I .EQ. J) GO TO 160
C
         DO 140 K = I, IGH
            ZR(I,K) = ZR(J,K)
            ZI(I,K) = ZI(J,K)
            ZR(J,K) = 0.0
            ZI(J,K) = 0.0
  140    CONTINUE
C
         ZR(J,I) = 1.0
  160 CONTINUE
```

```
C     ********** STORE ROOTS ISOLATED BY CBAL **********
  180 DO 200 I = 1, N
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 200
         WR(I) = HR(I,I)
         WI(I) = HI(I,I)
  200 CONTINUE
C
      EN = IGH
      TR = 0.0
      TI = 0.0
C     ********** SEARCH FOR NEXT EIGENVALUE **********
  220 IF (EN .LT. LOW) GO TO 680
      ITS = 0
      ENM1 = EN - 1
C     ********** LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW DO -- **********
  240 DO 260 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 300
         IF (ABS(HR(L,L-1)) + ABS(HI(L,L-1)) .LE.
     X       MACHEP * (ABS(HR(L-1,L-1)) + ABS(HI(L-1,L-1))
     X               + ABS(HR(L,L)) + ABS(HI(L,L)))) GO TO 300
  260 CONTINUE
C     ********** FORM SHIFT **********
  300 IF (L .EQ. EN) GO TO 660
      IF (ITS .EQ. 30) GO TO 1000
      IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
      SR = HR(EN,EN)
      SI = HI(EN,EN)
      XR = HR(ENM1,EN) * HR(EN,ENM1) - HI(ENM1,EN) * HI(EN,ENM1)
      XI = HR(ENM1,EN) * HI(EN,ENM1) + HI(ENM1,EN) * HR(EN,ENM1)
      IF (XR .EQ. 0.0 .AND. XI .EQ. 0.0) GO TO 340
      YR = (HR(ENM1,ENM1) - SR) / 2.0
      YI = (HI(ENM1,ENM1) - SI) / 2.0
      Z3 = CSQRT(CMPLX(YR**2-YI**2+XR,2.0*YR*YI+XI))
      ZZR = REAL(Z3)
      ZZI = AIMAG(Z3)
      IF (YR * ZZR + YI * ZZI .GE. 0.0) GO TO 310
      ZZR = -ZZR
      ZZI = -ZZI
  310 Z3 = CMPLX(XR,XI) / CMPLX(YR+ZZR,YI+ZZI)
      SR = SR - REAL(Z3)
      SI = SI - AIMAG(Z3)
      GO TO 340
C     ********** FORM EXCEPTIONAL SHIFT **********
  320 SR = ABS(HR(EN,ENM1)) + ABS(HR(ENM1,EN-2))
      SI = ABS(HI(EN,ENM1)) + ABS(HI(ENM1,EN-2))
C
  340 DO 360 I = LOW, EN
         HR(I,I) = HR(I,I) - SR
         HI(I,I) = HI(I,I) - SI
  360 CONTINUE
```

```
C
      TR = TR + SR
      TI = TI + SI
      ITS = ITS + 1
C     ********** LOOK FOR TWO CONSECUTIVE SMALL
C                SUB-DIAGONAL ELEMENTS **********
      XR = ABS(HR(ENM1,ENM1)) + ABS(HI(ENM1,ENM1))
      YR = ABS(HR(EN,ENM1)) + ABS(HI(EN,ENM1))
      ZZR = ABS(HR(EN,EN)) + ABS(HI(EN,EN))
C     ********** FOR M=EN-1 STEP -1 UNTIL L DO -- **********
      DO 380 MM = L, ENM1
         M = ENM1 + L - MM
         IF (M .EQ. L) GO TO 420
         YI = YR
         YR = ABS(HR(M,M-1)) + ABS(HI(M,M-1))
         XI = ZZR
         ZZR = XR
         XR = ABS(HR(M-1,M-1)) + ABS(HI(M-1,M-1))
         IF (YR .LE. MACHEP * ZZR / YI * (ZZR + XR + XI)) GO TO 420
  380 CONTINUE
C     ********** TRIANGULAR DECOMPOSITION H=L*R **********
  420 MP1 = M + 1
C
      DO 520 I = MP1, EN
         IM1 = I - 1
         XR = HR(IM1,IM1)
         XI = HI(IM1,IM1)
         YR = HR(I,IM1)
         YI = HI(I,IM1)
         IF (ABS(XR) + ABS(XI) .GE. ABS(YR) + ABS(YI)) GO TO 460
C     ********** INTERCHANGE ROWS OF HR AND HI **********
         DO 440 J = IM1, N
            ZZR = HR(IM1,J)
            HR(IM1,J) = HR(I,J)
            HR(I,J) = ZZR
            ZZI = HI(IM1,J)
            HI(IM1,J) = HI(I,J)
            HI(I,J) = ZZI
  440    CONTINUE
C
         Z3 = CMPLX(XR,XI) / CMPLX(YR,YI)
         WR(I) = 1.0
         GO TO 480
  460    Z3 = CMPLX(YR,YI) / CMPLX(XR,XI)
         WR(I) = -1.0
  480    ZZR = REAL(Z3)
         ZZI = AIMAG(Z3)
         HR(I,IM1) = ZZR
         HI(I,IM1) = ZZI
C
         DO 500 J = I, N
            HR(I,J) = HR(I,J) - ZZR * HR(IM1,J) + ZZI * HI(IM1,J)
            HI(I,J) = HI(I,J) - ZZR * HI(IM1,J) - ZZI * HR(IM1,J)
  500    CONTINUE
```

```
C
  520 CONTINUE
C     ********** COMPOSITION R*L=H **********
      DO 640 J = MP1, EN
         XR = HR(J,J-1)
         XI = HI(J,J-1)
         HR(J,J-1) = 0.0
         HI(J,J-1) = 0.0
C     ********** INTERCHANGE COLUMNS OF HR, HI, ZR, AND ZI,
C                IF NECESSARY **********
         IF (WR(J) .LE. 0.0) GO TO 580
C
         DO 540 I = 1, J
            ZZR = HR(I,J-1)
            HR(I,J-1) = HR(I,J)
            HR(I,J) = ZZR
            ZZI = HI(I,J-1)
            HI(I,J-1) = HI(I,J)
            HI(I,J) = ZZI
  540    CONTINUE
C
         DO 560 I = LOW, IGH
            ZZR = ZR(I,J-1)
            ZR(I,J-1) = ZR(I,J)
            ZR(I,J) = ZZR
            ZZI = ZI(I,J-1)
            ZI(I,J-1) = ZI(I,J)
            ZI(I,J) = ZZI
  560    CONTINUE
C
  580    DO 600 I = 1, J
            HR(I,J-1) = HR(I,J-1) + XR * HR(I,J) - XI * HI(I,J)
            HI(I,J-1) = HI(I,J-1) + XR * HI(I,J) + XI * HR(I,J)
  600    CONTINUE
C     ********** ACCUMULATE TRANSFORMATIONS **********
         DO 620 I = LOW, IGH
            ZR(I,J-1) = ZR(I,J-1) + XR * ZR(I,J) - XI * ZI(I,J)
            ZI(I,J-1) = ZI(I,J-1) + XR * ZI(I,J) + XI * ZR(I,J)
  620    CONTINUE
C
  640 CONTINUE
C
      GO TO 240
C     ********** A ROOT FOUND **********
  660 HR(EN,EN) = HR(EN,EN) + TR
      WR(EN) = HR(EN,EN)
      HI(EN,EN) = HI(EN,EN) + TI
      WI(EN) = HI(EN,EN)
      EN = ENM1
      GO TO 220
C     ********** ALL ROOTS FOUND.  BACKSUBSTITUTE TO FIND
C                VECTORS OF UPPER TRIANGULAR FORM **********
  680 NORM = 0.0
```

```
C
      DO 720 I = 1, N
C
         DO 720 J = I, N
            NORM = NORM + ABS(HR(I,J)) + ABS(HI(I,J))
  720 CONTINUE
C
      HR(1,1) = NORM
      IF (N .EQ. 1 .OR. NORM .EQ. 0.0) GO TO 1001
C     ********** FOR EN=N STEP -1 UNTIL 2 DO -- **********
      DO 800 NN = 2, N
         EN = N + 2 - NN
         XR = WR(EN)
         XI = WI(EN)
         ENM1 = EN - 1
C     ********** FOR I=EN-1 STEP -1 UNTIL 1 DO -- **********
         DO 780 II = 1, ENM1
            I = EN - II
            ZZR = HR(I,EN)
            ZZI = HI(I,EN)
            IF (I .EQ. ENM1) GO TO 760
            IP1 = I + 1
C
            DO 740 J = IP1, ENM1
               ZZR = ZZR + HR(I,J) * HR(J,EN) - HI(I,J) * HI(J,EN)
               ZZI = ZZI + HR(I,J) * HI(J,EN) + HI(I,J) * HR(J,EN)
  740       CONTINUE
C
  760       YR = XR - WR(I)
            YI = XI - WI(I)
            IF (YR .EQ. 0.0 .AND. YI .EQ. 0.0) YR = MACHEP * NORM
            Z3 = CMPLX(ZZR,ZZI) / CMPLX(YR,YI)
            HR(I,EN) = REAL(Z3)
            HI(I,EN) = AIMAG(Z3)
  780    CONTINUE
C
  800 CONTINUE
C     ********** END BACKSUBSTITUTION **********
      ENM1 = N - 1
C     ********** VECTORS OF ISOLATED ROOTS **********
      DO 840 I = 1, ENM1
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 840
         IP1 = I + 1
C
         DO 820 J = IP1, N
            ZR(I,J) = HR(I,J)
            ZI(I,J) = HI(I,J)
  820    CONTINUE
C
  840 CONTINUE
```

```
C      ********** MULTIPLY BY TRANSFORMATION MATRIX TO GIVE
C                 VECTORS OF ORIGINAL FULL MATRIX.
C                 FOR J=N STEP -1 UNTIL LOW+1 DO -- **********
       DO 880 JJ = LOW, ENM1
          J = N + LOW - JJ
          M = MIN0(J-1,IGH)
C
          DO 880 I = LOW, IGH
             ZZR = ZR(I,J)
             ZZI = ZI(I,J)
C
             DO 860 K = LOW, M
                ZZR = ZZR + ZR(I,K) * HR(K,J) - ZI(I,K) * HI(K,J)
                ZZI = ZZI + ZR(I,K) * HI(K,J) + ZI(I,K) * HR(K,J)
  860        CONTINUE
C
             ZR(I,J) = ZZR
             ZI(I,J) = ZZI
  880 CONTINUE
C
       GO TO 1001
C      ********** SET ERROR -- NO CONVERGENCE TO AN
C                 EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = EN
 1001 RETURN
       END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F245 COMQR

A Fortran IV Subroutine to Determine the Eigenvalues of a Complex Upper Hessenberg Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine COMQR computes the eigenvalues of a complex upper Hessenberg matrix using the QR method.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  COMQR(NM,N,LOW,IGH,HR,HI,WR,WI,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays HR and HI as specified in the DIMENSION statements for HR and HI in the calling program.

N is an integer input variable set equal to the order of the matrix H = (HR,HI). N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F271 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

HR,HI are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, HR and HI contain the real and imaginary parts, respectively, of the complex upper Hessenberg matrix. Note: COMQR destroys this upper Hessenberg matrix.

WR,WI are working precision real output one-dimensional variables of dimension at least N containing the real and imaginary parts, respectively, of the eigenvalues of the upper Hessenberg matrix.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, this subroutine terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the WR and WI arrays should be correct for indices IERR+1,IERR+2,...,N.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine some of the eigenvectors of a complex Hessenberg matrix, COMQR should be followed by CINVIT (F297) to compute those eigenvectors. Note, however, that the upper Hessenberg matrix must be saved before COMQR for later use of CINVIT, since COMQR destroys this upper Hessenberg matrix.

To determine the eigenvalues of a complex general matrix, COMQR should be preceded by CORTH (F244) to provide a suitable complex upper Hessenberg matrix for COMQR.

To determine some of the eigenvectors of a complex general matrix, COMQR should be preceded by CORTH (F244) and followed by CINVIT (F297) and CORTB (F247). CORTB back transforms the eigenvectors from CINVIT into those of the complex general matrix. Note, as above, that the Hessenberg matrix must be saved before COMQR.

It is recommended in general that CBAL (F271) be used before CORTH in which case CBABK2 (F272) must be used after CORTB if eigenvectors are computed.

In this implementation, the arithmetic is real throughout except for complex square root and complex division.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the QR method. The essence of this method is a process whereby a sequence of complex upper Hessenberg matrices, similar to the original Hessenberg matrix, is formed which converges to a triangular matrix. The rate of convergence of this sequence is improved by shifting the origin at each iteration. Before each iteration, the last Hessenberg form is checked for a possible splitting into submatrices. If a splitting occurs, only the lower submatrix participates in the next iteration.

The origin shift at each iteration is the eigenvalue of the lowest 2x2 principal minor closer to the second diagonal element of this minor. Whenever a lowest 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision.

The subdiagonal elements are rendered real initially by a diagonal unitary similarity transformation and maintained real throughout COMQR.

Some of the eigenvalues may have been isolated on the diagonal by the subroutine CBAL (F271). This information is transmitted to COMQR through the parameters LOW and IGH. As a result, COMQR immediately extracts the eigenvalues in rows 1 to LOW-1 and IGH+1 to N, and so applies the QR procedure to the submatrix situated in rows and columns LOW through IGH.

This subroutine is a translation of a unitary analogue of the Algol procedure COMLR written and discussed in detail by Martin and Wilkinson (1). The unitary analogue substitutes for the LR algorithm the QR algorithm, written and discussed in detail by Francis (2).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., The Modified LR Algorithm for Complex Hessenberg Matrices, Num. Math. 12,369-376 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/16, 396-403, Springer-Verlag, 1971.)

2) Francis, J.G.F., The QR Transformation - Parts 1 and 2, Comp. J. 4,265-271 and 332-345 (1961/62).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The subroutine COMQR is numerically stable (1,2); that is, each computed eigenvalue is exact for a matrix close to the original upper Hessenberg matrix.

```
      SUBROUTINE COMQR(NM,N,LOW,IGH,HR,HI,WR,WI,IERR)
C
      INTEGER I,J,L,N,EN,LL,NM,IGH,ITS,LOW,LP1,ENM1,IERR
      REAL HR(NM,N),HI(NM,N),WR(N),WI(N)
      REAL SI,SR,TI,TR,XI,XR,YI,YR,ZZI,ZZR,NORM,MACHEP
      REAL SQRT,CABS,ABS
      INTEGER MINO
      COMPLEX Z3
      COMPLEX CSQRT,CMPLX
      REAL REAL,AIMAG
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      IF (LOW .EQ. IGH) GO TO 180
C     ********** CREATE REAL SUBDIAGONAL ELEMENTS **********
      L = LOW + 1
C
      DO 170 I = L, IGH
         LL = MINO(I+1,IGH)
         IF (HI(I,I-1) .EQ. 0.0) GO TO 170
         NORM = CABS(CMPLX(HR(I,I-1),HI(I,I-1)))
         YR = HR(I,I-1) / NORM
         YI = HI(I,I-1) / NORM
         HR(I,I-1) = NORM
         HI(I,I-1) = 0.0
C
         DO 155 J = I, IGH
            SI = YR * HI(I,J) - YI * HR(I,J)
            HR(I,J) = YR * HR(I,J) + YI * HI(I,J)
            HI(I,J) = SI
  155    CONTINUE
C
         DO 160 J = LOW, LL
            SI = YR * HI(J,I) + YI * HR(J,I)
            HR(J,I) = YR * HR(J,I) - YI * HI(J,I)
            HI(J,I) = SI
  160    CONTINUE
C
  170 CONTINUE
C     ********** STORE ROOTS ISOLATED BY CBAL **********
  180 DO 200 I = 1, N
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 200
         WR(I) = HR(I,I)
         WI(I) = HI(I,I)
  200 CONTINUE
C
      EN = IGH
      TR = 0.0
      TI = 0.0
```

```
C     ********** SEARCH FOR NEXT EIGENVALUE **********
  220 IF (EN .LT. LOW) GO TO 1001
      ITS = 0
      ENM1 = EN - 1
C     ********** LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                  FOR L=EN STEP -1 UNTIL LOW  -- **********
  240 DO 260 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 300
         IF (ABS(HR(L,L-1)) .LE.
     X       MACHEP * (ABS(HR(L-1,L-1)) + ABS(HI(L-1,L-1))
     X              + ABS(HR(L,L)) +ABS(HI(L,L)))) GO TO 300
  260 CONTINUE
C     ********** FORM SHIFT **********
  300 IF (L .EQ. EN) GO TO 660
      IF (ITS .EQ. 30) GO TO 1000
      IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
      SR = HR(EN,EN)
      SI = HI(EN,EN)
      XR = HR(ENM1,EN) * HR(EN,ENM1)
      XI = HI(ENM1,EN) * HR(EN,ENM1)
      IF (XR .EQ. 0.0 .AND. XI .EQ. 0.0) GO TO 340
      YR = (HR(ENM1,ENM1) - SR) / 2.0
      YI = (HI(ENM1,ENM1) - SI) / 2.0
      Z3 = CSQRT(CMPLX(YR**2-YI**2+XR,2.0*YR*YI+XI))
      ZZR = REAL(Z3)
      ZZI = AIMAG(Z3)
      IF (YR * ZZR + YI * ZZI .GE. 0.0) GO TO 310
      ZZR = -ZZR
      ZZI = -ZZI
  310 Z3 = CMPLX(XR,XI) / CMPLX(YR+ZZR,YI+ZZI)
      SR = SR - REAL(Z3)
      SI = SI - AIMAG(Z3)
      GO TO 340
C     ********** FORM EXCEPTIONAL SHIFT **********
  320 SR = ABS(HR(EN,ENM1)) + ABS(HR(ENM1,EN-2))
      SI = 0.0
C
  340 DO 360 I = LOW, EN
         HR(I,I) = HR(I,I) - SR
         HI(I,I) = HI(I,I) - SI
  360 CONTINUE
C
      TR = TR + SR
      TI = TI + SI
      ITS = ITS + 1
C     ********** REDUCE TO TRIANGLE (ROWS) **********
      LP1 = L + 1
```

```
C
      DO 500 I = LP1, EN
         SR = HR(I,I-1)
         HR(I,I-1) = 0.0
         NORM = SQRT(HR(I-1,I-1)*HR(I-1,I-1)+HI(I-1,I-1)*HI(I-1,I-1)
     X                +SR*SR)
         XR = HR(I-1,I-1) / NORM
         WR(I-1) = XR
         XI = HI(I-1,I-1) / NORM
         WI(I-1) = XI
         HR(I-1,I-1) = NORM
         HI(I-1,I-1) = 0.0
         HI(I,I-1) = SR / NORM
C
         DO 490 J = I, EN
            YR = HR(I-1,J)
            YI = HI(I-1,J)
            ZZR = HR(I,J)
            ZZI = HI(I,J)
            HR(I-1,J) = XR * YR + XI * YI + HI(I,I-1) * ZZR
            HI(I-1,J) = XR * YI - XI * YR + HI(I,I-1) * ZZI
            HR(I,J) = XR * ZZR - XI * ZZI - HI(I,I-1) * YR
            HI(I,J) = XR * ZZI + XI * ZZR - HI(I,I-1) * YI
  490    CONTINUE
C
  500 CONTINUE
C
      SI = HI(EN,EN)
      IF (SI .EQ. 0.0) GO TO 540
      NORM = CABS(CMPLX(HR(EN,EN),SI))
      SR = HR(EN,EN) / NORM
      SI = SI / NORM
      HR(EN,EN) = NORM
      HI(EN,EN) = 0.0
C     ********** INVERSE OPERATION (COLUMNS) **********
  540 DO 600 J = LP1, EN
         XR = WR(J-1)
         XI = WI(J-1)
C
         DO 580 I = L, J
            YR = HR(I,J-1)
            YI = 0.0
            ZZR = HR(I,J)
            ZZI = HI(I,J)
            IF (I .EQ. J) GO TO 560
            YI = HI(I,J-1)
            HI(I,J-1) = XR * YI + XI * YR + HI(J,J-1) * ZZI
  560       HR(I,J-1) = XR * YR - XI * YI + HI(J,J-1) * ZZR
            HR(I,J) = XR * ZZR + XI * ZZI - HI(J,J-1) * YR
            HI(I,J) = XR * ZZI - XI * ZZR - HI(J,J-1) * YI
  580    CONTINUE
C
  600 CONTINUE
```

```
C
      IF (SI .EQ. 0.0) GO TO 240
C
      DO 630 I = L, EN
         YR = HR(I,EN)
         YI = HI(I,EN)
         HR(I,EN) = SR * YR - SI * YI
         HI(I,EN) = SR * YI + SI * YR
  630 CONTINUE
C
      GO TO 240
C     ********** A ROOT FOUND **********
  660 WR(EN) = HR(EN,EN) + TR
      WI(EN) = HI(EN,EN) + TI
      EN = ENM1
      GO TO 220
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = EN
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F246 COMQR2

A Fortran IV Subroutine to Determine the Eigenvalues and Eigenvectors of a Complex Upper Hessenberg Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine COMQR2 computes the eigenvalues and eigenvectors of a complex upper Hessenberg matrix. COMQR2 uses the QR method to compute the eigenvalues and accumulates the QR transformations to compute the eigenvectors. The eigenvectors of a complex general matrix can also be computed directly by COMQR2, if CORTH (F244) has been used to reduce this matrix to Hessenberg form.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  COMQR2(NM,N,LOW,IGH,ORTR,ORTI,HR,HI,
                   WR,WI,ZR,ZI,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays HR, HI, ZR, and ZI as specified in the DIMENSION statements for HR, HI, ZR, and ZI in the calling program.

N is an integer input variable set equal to the order of the matrix H = (HR,HI). N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F271 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

ORTR,ORTI
: are working precision real one-dimensional variables of dimension at least IGH. Only components LOW through IGH are used by COMQR2. If the eigenvectors of the Hessenberg matrix are desired, set ORTR(J)=ORTI(J)=0.0 for J = LOW,LOW+1,...,IGH. If the eigenvectors of a complex general matrix are desired, ORTR and ORTI contain some information about the unitary transformations generated by CORTH during the reduction to the Hessenberg form.

HR,HI
: are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, HR and HI contain the real and imaginary parts, respectively, of the complex upper Hessenberg matrix. If the eigenvectors of the Hessenberg matrix are desired, the lower triangles of HR,HI below the subdiagonal may be arbitrary. If the eigenvectors of a complex general matrix are desired, the lower triangles of HR,HI contain further information about the unitary transformations generated by CORTH in the reduction to the Hessenberg form. Note: COMQR2 destroys the upper Hessenberg portions of HR,HI.

WR,WI
: are working precision real output one-dimensional variables of dimension at least N containing the real and imaginary parts, respectively, of the eigenvalues of the upper Hessenberg matrix.

ZR,ZI
: are working precision real output two-dimensional variables with row dimension NM and column dimension at least N containing the real and imaginary parts, respectively, of the eigenvectors. The eigenvectors are not normalized.

IERR
: is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, this subroutine terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the WR and WI arrays should be correct for indices IERR+1,IERR+2,...,N, but no eigenvectors are computed.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine the eigenvalues and eigenvectors of a complex general matrix, COMQR2 must be preceded by CORTH (F244) to provide a suitable complex Hessenberg matrix for COMQR2.

It is recommended in general that CBAL (F271) be used before CORTH in which case CBABK2 (F272) must be used after COMQR2.

In this implementation, the arithmetic is real throughout except for complex square root and complex division.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the QR method. The essence of this method is a process whereby a sequence of complex upper Hessenberg matrices, similar to the original Hessenberg matrix, is formed which converges to a triangular matrix. The rate of convergence of this sequence is improved by shifting the origin at each iteration. Before each iteration, the last Hessenberg form is checked for a possible splitting into submatrices. If a splitting occurs, only the lower submatrix participates in the next iteration. The similarity transformations used in each iteration are accumulated in the ZR and ZI arrays along with the reduction transformations for the original matrix.

The origin shift at each iteration is the eigenvalue of the lowest 2x2 principal minor closer to the second diagonal element of this minor. Whenever a lowest 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the original matrix and the algorithm proceeds with the remaining submatrix.

This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision. The eigenvectors of this triangular matrix are determined by back substitution, and then back transformed into those of the original matrix.

The subdiagonal elements are rendered real initially by a diagonal unitary similarity transformation and maintained real throughout COMQR2.

Some of the eigenvalues may have been isolated on the diagonal by the subroutine CBAL (F271). This information is transmitted to COMQR2 through the parameters LOW and IGH. As a result, COMQR2 immediately extracts the eigenvalues in rows 1 to LOW-1 and IGH+1 to N, and so applies the QR procedure to the submatrix situated in rows and columns LOW through IGH.

This subroutine is a translation of a unitary analogue of the Algol procedure COMLR2 written and discussed in detail by Peters and Wilkinson (1). The unitary analogue substitutes for the LR algorithm the QR algorithm, written and discussed in detail by Francis (2).

4. REFERENCES.

1) Peters, G. and Wilkinson, J.H., Eigenvectors of Real and Complex Matrices by LR and QR Triangularizations, Num. Math. 16,181-204 (1970). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/15, 372-395, Springer-Verlag, 1971.)

2) Francis, J.G.F., The QR Transformation - Parts 1 and 2, Comp. J. 4,265-271 and 332-345 (1961/62).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The subroutine COMQR2 is numerically stable (1,2); that is, each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original upper Hessenberg matrix.

```
      SUBROUTINE COMQR2(NM,N,LOW,IGH,ORTR,ORTI,HR,HI,WR,WI,ZR,ZI,IERR)
C
      INTEGER I,J,K,L,M,N,EN,II,JJ,LL,NM,NN,IGH,IP1,
     X        ITS,LOW,LP1,ENM1,IEND,IERR
      REAL HR(NM,N),HI(NM,N),WR(N),WI(N),ZR(NM,N),ZI(NM,N),
     X       ORTR(IGH),ORTI(IGH)
      REAL SI,SR,TI,TR,XI,XR,YI,YR,ZZI,ZZR,NORM,MACHEP
      REAL SQRT,CABS,ABS
      INTEGER MINO
      COMPLEX Z3
      COMPLEX CSQRT,CMPLX
      REAL REAL,AIMAG
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
C     ********** INITIALIZE EIGENVECTOR MATRIX **********
      DO 100 I = 1, N
C
         DO 100 J = 1, N
            ZR(I,J) = 0.0
            ZI(I,J) = 0.0
            IF (I .EQ. J) ZR(I,J) = 1.0
  100 CONTINUE
C     ********** FORM THE MATRIX OF ACCUMULATED TRANSFORMATIONS
C                FROM THE INFORMATION LEFT BY CORTH **********
      IEND = IGH - LOW - 1
      IF (IEND) 180, 150, 105
C     ********** FOR I=IGH-1 STEP -1 UNTIL LOW+1 DO -- **********
  105 DO 140 II = 1, IEND
         I = IGH - II
         IF (ORTR(I) .EQ. 0.0 .AND. ORTI(I) .EQ. 0.0) GO TO 140
         IF (HR(I,I-1) .EQ. 0.0 .AND. HI(I,I-1) .EQ. 0.0) GO TO 140
C     ********** NORM BELOW IS NEGATIVE OF H FORMED IN CORTH **********
         NORM = HR(I,I-1) * ORTR(I) + HI(I,I-1) * ORTI(I)
         IP1 = I + 1
C
         DO 110 K = IP1, IGH
            ORTR(K) = HR(K,I-1)
            ORTI(K) = HI(K,I-1)
  110    CONTINUE
C
         DO 130 J = I, IGH
            SR = 0.0
            SI = 0.0
C
            DO 115 K = I, IGH
               SR = SR + ORTR(K) * ZR(K,J) + ORTI(K) * ZI(K,J)
               SI = SI + ORTR(K) * ZI(K,J) - ORTI(K) * ZR(K,J)
  115       CONTINUE
```

```
C
            SR = SR / NORM
            SI = SI / NORM
C
            DO 120 K = I, IGH
               ZR(K,J) = ZR(K,J) + SR * ORTR(K) - SI * ORTI(K)
               ZI(K,J) = ZI(K,J) + SR * ORTI(K) + SI * ORTR(K)
  120       CONTINUE
C
  130    CONTINUE
C
  140 CONTINUE
C     ********** CREATE REAL SUBDIAGONAL ELEMENTS **********
  150 L = LOW + 1
C
      DO 170 I = L, IGH
         LL = MIN0(I+1,IGH)
         IF (HI(I,I-1) .EQ. 0.0) GO TO 170
         NORM = CABS(CMPLX(HR(I,I-1),HI(I,I-1)))
         YR = HR(I,I-1) / NORM
         YI = HI(I,I-1) / NORM
         HR(I,I-1) = NORM
         HI(I,I-1) = 0.0
C
         DO 155 J = I, N
            SI = YR * HI(I,J) - YI * HR(I,J)
            HR(I,J) = YR * HR(I,J) + YI * HI(I,J)
            HI(I,J) = SI
  155    CONTINUE
C
         DO 160 J = 1, LL
            SI = YR * HI(J,I) + YI * HR(J,I)
            HR(J,I) = YR * HR(J,I) - YI * HI(J,I)
            HI(J,I) = SI
  160    CONTINUE
C
         DO 165 J = LOW, IGH
            SI = YR * ZI(J,I) + YI * ZR(J,I)
            ZR(J,I) = YR * ZR(J,I) - YI * ZI(J,I)
            ZI(J,I) = SI
  165    CONTINUE
C
  170 CONTINUE
C     ********** STORE ROOTS ISOLATED BY CBAL **********
  180 DO 200 I = 1, N
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 200
         WR(I) = HR(I,I)
         WI(I) = HI(I,I)
  200 CONTINUE
C
      EN = IGH
      TR = 0.0
      TI = 0.0
```

```
C     ********** SEARCH FOR NEXT EIGENVALUE **********
  220 IF (EN .LT. LOW) GO TO 680
      ITS = 0
      ENM1 = EN - 1
C     ********** LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW DO -- **********
  240 DO 260 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 300
         IF (ABS(HR(L,L-1)) .LE.
     X       MACHEP * (ABS(HR(L-1,L-1)) + ABS(HI(L-1,L-1))
     X             + ABS(HR(L,L)) +ABS(HI(L,L)))) GO TO 300
  260 CONTINUE
C     ********** FORM SHIFT **********
  300 IF (L .EQ. EN) GO TO 660
      IF (ITS .EQ. 30) GO TO 1000
      IF (ITS .EQ. 10 .OR. ITS .EQ. 20) GO TO 320
      SR = HR(EN,EN)
      SI = HI(EN,EN)
      XR = HR(ENM1,EN) * HR(EN,ENM1)
      XI = HI(ENM1,EN) * HR(EN,ENM1)
      IF (XR .EQ. 0.0 .AND. XI .EQ. 0.0) GO TO 340
      YR = (HR(ENM1,ENM1) - SR) / 2.0
      YI = (HI(ENM1,ENM1) - SI) / 2.0
      Z3 = CSQRT(CMPLX(YR**2-YI**2+XR,2.0*YR*YI+XI))
      ZZR = REAL(Z3)
      ZZI = AIMAG(Z3)
      IF (YR * ZZR + YI * ZZI .GE. 0.0) GO TO 310
      ZZR = -ZZR
      ZZI = -ZZI
  310 Z3 = CMPLX(XR,XI) / CMPLX(YR+ZZR,YI+ZZI)
      SR = SR - REAL(Z3)
      SI = SI - AIMAG(Z3)
      GO TO 340
C     ********** FORM EXCEPTIONAL SHIFT **********
  320 SR = ABS(HR(EN,ENM1)) + ABS(HR(ENM1,EN-2))
      SI = 0.0
C
  340 DO 360 I = LOW, EN
         HR(I,I) = HR(I,I) - SR
         HI(I,I) = HI(I,I) - SI
  360 CONTINUE
C
      TR = TR + SR
      TI = TI + SI
      ITS = ITS + 1
C     ********** REDUCE TO TRIANGLE (ROWS) **********
      LP1 = L + 1
```

```
C
      DO 500 I = LP1, EN
         SR = HR(I,I-1)
         HR(I,I-1) = 0.0
         NORM = SQRT(HR(I-1,I-1)*HR(I-1,I-1)+HI(I-1,I-1)*HI(I-1,I-1)
     X                +SR*SR)
         XR = HR(I-1,I-1) / NORM
         WR(I-1) = XR
         XI = HI(I-1,I-1) / NORM
         WI(I-1) = XI
         HR(I-1,I-1) = NORM
         HI(I-1,I-1) = 0.0
         HI(I,I-1) = SR / NORM
C
         DO 490 J = I, N
            YR = HR(I-1,J)
            YI = HI(I-1,J)
            ZZR = HR(I,J)
            ZZI = HI(I,J)
            HR(I-1,J) = XR * YR + XI * YI + HI(I,I-1) * ZZR
            HI(I-1,J) = XR * YI - XI * YR + HI(I,I-1) * ZZI
            HR(I,J) = XR * ZZR - XI * ZZI - HI(I,I-1) * YR
            HI(I,J) = XR * ZZI + XI * ZZR - HI(I,I-1) * YI
  490    CONTINUE
C
  500 CONTINUE
C
      SI = HI(EN,EN)
      IF (SI .EQ. 0.0) GO TO 540
      NORM = CABS(CMPLX(HR(EN,EN),SI))
      SR = HR(EN,EN) / NORM
      SI = SI / NORM
      HR(EN,EN) = NORM
      HI(EN,EN) = 0.0
      IF (EN .EQ. N) GO TO 540
      IP1 = EN + 1
C
      DO 520 J = IP1, N
         YR = HR(EN,J)
         YI = HI(EN,J)
         HR(EN,J) = SR * YR + SI * YI
         HI(EN,J) = SR * YI - SI * YR
  520 CONTINUE
C     ********** INVERSE OPERATION (COLUMNS) **********
  540 DO 600 J = LP1, EN
         XR = WR(J-1)
         XI = WI(J-1)
C
         DO 580 I = 1, J
            YR = HR(I,J-1)
            YI = 0.0
            ZZR = HR(I,J)
            ZZI = HI(I,J)
            IF (I .EQ. J) GO TO 560
```

```
               YI = HI(I,J-1)
               HI(I,J-1) = XR * YI + XI * YR + HI(J,J-1) * ZZI
  560          HR(I,J-1) = XR * YR - XI * YI + HI(J,J-1) * ZZR
               HR(I,J) = XR * ZZR + XI * ZZI - HI(J,J-1) * YR
               HI(I,J) = XR * ZZI - XI * ZZR - HI(J,J-1) * YI
  580       CONTINUE
C
            DO 590 I = LOW, IGH
               YR = ZR(I,J-1)
               YI = ZI(I,J-1)
               ZZR = ZR(I,J)
               ZZI = ZI(I,J)
               ZR(I,J-1) = XR * YR - XI * YI + HI(J,J-1) * ZZR
               ZI(I,J-1) = XR * YI + XI * YR + HI(J,J-1) * ZZI
               ZR(I,J) = XR * ZZR + XI * ZZI - HI(J,J-1) * YR
               ZI(I,J) = XR * ZZI - XI * ZZR - HI(J,J-1) * YI
  590       CONTINUE
C
  600    CONTINUE
C
         IF (SI .EQ. 0.0) GO TO 240
C
         DO 630 I = 1, EN
            YR = HR(I,EN)
            YI = HI(I,EN)
            HR(I,EN) = SR * YR - SI * YI
            HI(I,EN) = SR * YI + SI * YR
  630    CONTINUE
C
         DO 640 I = LOW, IGH
            YR = ZR(I,EN)
            YI = ZI(I,EN)
            ZR(I,EN) = SR * YR - SI * YI
            ZI(I,EN) = SR * YI + SI * YR
  640    CONTINUE
C
         GO TO 240
C        ********** A ROOT FOUND **********
  660    HR(EN,EN) = HR(EN,EN) + TR
         WR(EN) = HR(EN,EN)
         HI(EN,EN) = HI(EN,EN) + TI
         WI(EN) = HI(EN,EN)
         EN = ENM1
         GO TO 220
C        ********** ALL ROOTS FOUND.  BACKSUBSTITUTE TO FIND
C                   VECTORS OF UPPER TRIANGULAR FORM **********
  680    NORM = 0.0
C
         DO 720 I = 1, N
C
            DO 720 J = I, N
               NORM = NORM + ABS(HR(I,J)) + ABS(HI(I,J))
  720    CONTINUE
```

```
C
      IF (N .EQ. 1 .OR. NORM .EQ. 0.0) GO TO 1001
C     ********** FOR EN=N STEP -1 UNTIL 2 DO -- **********
      DO 800 NN = 2, N
         EN = N + 2 - NN
         XR = WR(EN)
         XI = WI(EN)
         ENM1 = EN - 1
C     ********** FOR I=EN-1 STEP -1 UNTIL 1 DO -- **********
         DO 780 II = 1, ENM1
            I = EN - II
            ZZR = HR(I,EN)
            ZZI = HI(I,EN)
            IF (I .EQ. ENM1) GO TO 760
            IP1 = I + 1
C
            DO 740 J = IP1, ENM1
               ZZR = ZZR + HR(I,J) * HR(J,EN) - HI(I,J) * HI(J,EN)
               ZZI = ZZI + HR(I,J) * HI(J,EN) + HI(I,J) * HR(J,EN)
  740       CONTINUE
C
  760       YR = XR - WR(I)
            YI = XI - WI(I)
            IF (YR .EQ. 0.0 .AND. YI .EQ. 0.0) YR = MACHEP * NORM
            Z3 = CMPLX(ZZR,ZZI) / CMPLX(YR,YI)
            HR(I,EN) = REAL(Z3)
            HI(I,EN) = AIMAG(Z3)
  780    CONTINUE
C
  800 CONTINUE
C     ********** END BACKSUBSTITUTION **********
      ENM1 = N - 1
C     ********** VECTORS OF ISOLATED ROOTS **********
      DO  840 I = 1, ENM1
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 840
         IP1 = I + 1
C
         DO 820 J = IP1, N
            ZR(I,J) = HR(I,J)
            ZI(I,J) = HI(I,J)
  820    CONTINUE
C
  840 CONTINUE
C     ********** MULTIPLY BY TRANSFORMATION MATRIX TO GIVE
C                VECTORS OF ORIGINAL FULL MATRIX.
C                FOR J=N STEP -1 UNTIL LOW+1 DO -- **********
      DO 880 JJ = LOW, ENM1
         J = N + LOW - JJ
         M = MIN0(J-1,IGH)
C
         DO 880 I = LOW, IGH
            ZZR = ZR(I,J)
            ZZI = ZI(I,J)
```

```
C
            DO 860 K = LOW, M
               ZZR = ZZR + ZR(I,K) * HR(K,J) - ZI(I,K) * HI(K,J)
               ZZI = ZZI + ZR(I,K) * HI(K,J) + ZI(I,K) * HR(K,J)
  860       CONTINUE
C
            ZR(I,J) = ZZR
            ZI(I,J) = ZZI
  880 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = EN
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F247 CORTB

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Upper Hessenberg Matrix Determined by CORTH.

July, 1975

1. PURPOSE.

The Fortran IV subroutine CORTB forms the eigenvectors of a complex general matrix from the eigenvectors of that upper Hessenberg matrix determined by CORTH (F244).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  CORTB(NM,LOW,IGH,AR,AI,ORTR,ORTI,
                              M,ZR,ZI)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays AR, AI, ZR, and ZI as specified in the DIMENSION statements for AR, AI, ZR, and ZI in the calling program.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F271 for the details. If the matrix is not balanced, set LOW to 1 and IGH to the order of the matrix.

AR,AI are working precision real input two-dimensional variables with row dimension NM and column dimension at least IGH. Their lower triangles contain some information about the unitary transformations used in

the reduction to the Hessenberg form. The remaining upper parts of AR and AI are arbitrary. See section 3 of F244 for the details.

ORTR,ORTI
: are working precision real one-dimensional variables of dimension at least IGH. On input, ORTR and ORTI contain further information about the unitary transformations used in the reduction by CORTH. See section 3 of F244 for the details. ORTR and ORTI are used for temporary storage within CORTB and are not restored.

M
: is an integer input variable set equal to the number of columns of Z = (ZR,ZI) to be back transformed.

ZR,ZI
: are working precision real two-dimensional variables with row dimension NM and column dimension at least M. On input, the first M columns of ZR and ZI contain the real and imaginary parts, respectively, of the eigenvectors to be back transformed. On output, these M columns of ZR and ZI contain the real and imaginary parts of the transformed eigenvectors.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine CORTH (F244).

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix C (say) has been reduced to the upper Hessenberg form F stored in A by the similarity transformation

$$F = Q^{*} C Q$$

where Q is a product of the unitary matrices encoded in ORTR,ORTI and in a lower triangle of A under F. Then, given an array Z of column vectors, CORTB computes the matrix product QZ. If the eigenvectors of F are columns of the array Z, then CORTB forms the eigenvectors of C in their place.

This subroutine is a translation of a complex analogue of the Algol procedure ORTBAK written and discussed in detail by Martin and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., Similarity Reduction of a General Matrix to Hessenberg Form, Num. Math. 12,349-368 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/13, 339-358, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The accuracy of CORTB can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of complex general matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

```
      SUBROUTINE CORTB(NM,LOW,IGH,AR,AI,ORTR,ORTI,M,ZR,ZI)
C
      INTEGER I,J,M,LA,MM,MP,NM,IGH,KP1,LOW,MP1
      REAL AR(NM,IGH),AI(NM,IGH),ORTR(IGH),ORTI(IGH),ZR(NM,M),ZI(NM,M)
      REAL H,GI,GR
C
      IF (M .EQ. 0) GO TO 200
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C     ********** FOR MP=IGH-1 STEP -1 UNTIL LOW+1 DO -- **********
      DO 140 MM = KP1, LA
         MP = LOW + IGH - MM
         IF (AR(MP,MP-1) .EQ. 0.0 .AND. AI(MP,MP-1) .EQ. 0.0)
     X      GO TO 140
C     ********** H BELOW IS NEGATIVE OF H FORMED IN CORTH **********
         H = AR(MP,MP-1) * ORTR(MP) + AI(MP,MP-1) * ORTI(MP)
         MP1 = MP + 1
C
         DO 100 I = MP1, IGH
            ORTR(I) = AR(I,MP-1)
            ORTI(I) = AI(I,MP-1)
  100    CONTINUE
C
         DO 130 J = 1, M
            GR = 0.0
            GI = 0.0
C
            DO 110 I = MP, IGH
               GR = GR + ORTR(I) * ZR(I,J) + ORTI(I) * ZI(I,J)
               GI = GI + ORTR(I) * ZI(I,J) - ORTI(I) * ZR(I,J)
  110       CONTINUE
C
            GR = GR / H
            GI = GI / H
C
            DO 120 I = MP, IGH
               ZR(I,J) = ZR(I,J) + GR * ORTR(I) - GI * ORTI(I)
               ZI(I,J) = ZI(I,J) + GR * ORTI(I) + GI * ORTR(I)
  120       CONTINUE
C
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F244 CORTH

A Fortran IV Subroutine to Reduce a Complex General Matrix to Upper Hessenberg Form Using Unitary Transformations.

July, 1975

1. PURPOSE.

The Fortran IV subroutine CORTH reduces a complex general matrix to complex upper Hessenberg form using unitary similarity transformations. This reduced form is used by other subroutines to find the eigenvalues and/or eigenvectors of the original matrix. See section 2C for the specific routines.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  CORTH(NM,N,LOW,IGH,AR,AI,ORTR,ORTI)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays AR and AI as specified in the DIMENSION statements for AR and AI in the calling program.

N is an integer input variable set equal to the order of the matrix A = (AR,AI). N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F271 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

AR,AI are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, AR and AI contain the real and imaginary parts, respectively, of the complex matrix of order N to be reduced to Hessenberg form. On output, AR and AI contain the real and imaginary parts of the upper Hessenberg matrix as well as some information about the unitary transformations used in the reduction. See section 3 for the details.

ORTR,ORTI
are working precision real output one-dimensional variables of dimension at least IGH containing the remaining information about the unitary transformations. See section 3 for the details. Only components LOW+1 through IGH are actually used by CORTH.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues of the original matrix are desired, this subroutine should be followed by COMQR (F245).

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by COMQR2 (F246).

If some of the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by COMQR (F245), CINVIT (F297), and CORTB (F247).

If the matrix has elements of widely varying magnitudes, the larger ones should be in the top left-hand corner.

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix A = (AR,AI) has the form

$$A = \begin{pmatrix} T & X & Y \\ 0 & B & Z \\ 0 & 0 & R \end{pmatrix}$$

where T and R are upper triangular matrices, B is a square matrix situated in rows and columns LOW through IGH, and X, Y, and Z are rectangular matrices of the appropriate dimensions. Then the subroutine CORTH performs unitary similarity transformations to reduce B to Hessenberg form.

The Hessenberg reduction is performed in the following way. Starting with J=LOW, the elements in the J-th column below the diagonal are first scaled, to avoid possible underflow in the transformation that might result in severe departure from orthogonality. The sum of squared magnitudes SIGMA of these scaled elements is next formed. Then, a vector U and a scalar

$$H = U^{*} U/2$$

define an operator

$$P = I - UU^{*}/H$$

which is unitary and Hermitian and for which the similarity transformation PAP eliminates the elements in the J-th column of A below the subdiagonal.

The non-zero components of U are the elements of the J-th column below the diagonal with the first of them augmented by the square root of SIGMA times the subdiagonal element divided by its magnitude. By saving this component in the array ORTR,ORTI and not overwriting the column elements eliminated in the transformation, full information on P is saved for later use in COMQR2 and CORTB.

The transformation replaces the subdiagonal element with the square root of SIGMA times the negative of the replaced element divided by its magnitude.

The above steps are repeated on further columns of the transformed A until B is reduced to Hessenberg form; that is, repeated for J = LOW+1,LOW+2,...,IGH-2.

This subroutine is a translation of a complex analogue of the Algol procedure ORTHES written and discussed in detail by Martin and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., Similarity Reduction of a General Matrix to Hessenberg Form, Num. Math. 12,349-368 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/13, 339-358, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex general matrices.

B. Accuracy.

The accuracy of CORTH can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of complex general matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

```
      SUBROUTINE CORTH(NM,N,LOW,IGH,AR,AI,ORTR,ORTI)
C
      INTEGER I,J,M,N,II,JJ,LA,MP,NM,IGH,KP1,LOW
      REAL AR(NM,N),AI(NM,N),ORTR(IGH),ORTI(IGH)
      REAL F,G,H,FI,FR,SCALE
      REAL SQRT,CABS,ABS
      COMPLEX CMPLX
C
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C
      DO 180 M = KP1, LA
         H = 0.0
         ORTR(M) = 0.0
         ORTI(M) = 0.0
         SCALE = 0.0
C     ********** SCALE COLUMN (ALGOL TOL THEN NOT NEEDED) **********
         DO 90 I = M, IGH
   90    SCALE = SCALE + ABS(AR(I,M-1)) + ABS(AI(I,M-1))
C
         IF (SCALE .EQ. 0.0) GO TO 180
         MP = M + IGH
C     ********** FOR I=IGH STEP -1 UNTIL M DO -- **********
         DO 100 II = M, IGH
            I = MP - II
            ORTR(I) = AR(I,M-1) / SCALE
            ORTI(I) = AI(I,M-1) / SCALE
            H = H + ORTR(I) * ORTR(I) + ORTI(I) * ORTI(I)
  100    CONTINUE
C
         G = SQRT(H)
         F = CABS(CMPLX(ORTR(M),ORTI(M)))
         IF (F .EQ. 0.0) GO TO 103
         H = H + F * G
         G = G / F
         ORTR(M) = (1.0 + G) * ORTR(M)
         ORTI(M) = (1.0 + G) * ORTI(M)
         GO TO 105
C
  103    ORTR(M) = G
         AR(M,M-1) = SCALE
C     ********** FORM (I-(U*UT)/H) * A **********
  105    DO 130 J = M, N
            FR = 0.0
            FI = 0.0
C     ********** FOR I=IGH STEP -1 UNTIL M DO -- **********
            DO 110 II = M, IGH
               I = MP - II
               FR = FR + ORTR(I) * AR(I,J) + ORTI(I) * AI(I,J)
               FI = FI + ORTR(I) * AI(I,J) - ORTI(I) * AR(I,J)
  110       CONTINUE
```

```
C
            FR = FR / H
            FI = FI / H
C
            DO 120 I = M, IGH
               AR(I,J) = AR(I,J) - FR * ORTR(I) + FI * ORTI(I)
               AI(I,J) = AI(I,J) - FR * ORTI(I) - FI * ORTR(I)
  120       CONTINUE
C
  130    CONTINUE
C     ********** FORM (I-(U*UT)/H)*A*(I-(U*UT)/H) **********
         DO 160 I = 1, IGH
            FR = 0.0
            FI = 0.0
C     ********** FOR J=IGH STEP -1 UNTIL M DO -- **********
            DO 140 JJ = M, IGH
               J = MP - JJ
               FR = FR + ORTR(J) * AR(I,J) - ORTI(J) * AI(I,J)
               FI = FI + ORTR(J) * AI(I,J) + ORTI(J) * AR(I,J)
  140       CONTINUE
C
            FR = FR / H
            FI = FI / H
C
            DO 150 J = M, IGH
               AR(I,J) = AR(I,J) - FR * ORTR(J) - FI * ORTI(J)
               AI(I,J) = AI(I,J) + FR * ORTI(J) - FI * ORTR(J)
  150       CONTINUE
C
  160    CONTINUE
C
         ORTR(M) = SCALE * ORTR(M)
         ORTI(M) = SCALE * ORTI(M)
         AR(M,M-1) = -G * AR(M,M-1)
         AI(M,M-1) = -G * AI(M,M-1)
  180 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F274-2 ELMBAK

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Upper Hessenberg Matrix Determined by ELMHES.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine ELMBAK forms the eigenvectors of a real general matrix from the eigenvectors of that upper Hessenberg matrix determined by ELMHES (F273).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE ELMBAK(NM,LOW,IGH,A,INT,M,Z)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F269 for the details. If the matrix is not balanced, set LOW to 1 and IGH to the order of the matrix.

A is a working precision real input two-dimensional variable with row dimension NM and column dimension at least IGH. Its lower triangle below the subdiagonal contains the multipliers which were used in the reduction to the Hessenberg form. The remaining upper part of A is arbitrary. See section 3 of F273 for the details.

INT is an integer input one-dimensional variable of dimension at least IGH. INT identifies the rows and columns interchanged during the reduction by ELMHES. See section 3 of F273 for the details.

M is an integer input variable set equal to the number of columns of Z to be back transformed.

Z is a working precision real two-dimensional variable with row dimension NM and column dimension at least M. On input, the first M columns of Z contain the real and imaginary parts of the eigenvectors to be back transformed. On output, these M columns of Z contain the real and imaginary parts of the transformed eigenvectors.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine ELMHES (F273).

The real and imaginary parts of an eigenvector need not be stored in consecutive columns of Z.

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix C (say) has been reduced to the upper Hessenberg form F stored in A by the similarity transformation

$$F = G^{-1} C G$$

where G is a product of the permutation and elementary matrices encoded in INT and in a lower triangle of A under F respectively. Then, given an array Z of column vectors, ELMBAK computes the matrix product GZ. If the real and imaginary parts of the eigenvectors of F are columns of the array Z, then ELMBAK forms the eigenvectors of C in their place.

This subroutine is a translation of the Algol procedure ELMBAK written and discussed in detail by Martin and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., Similarity Reduction of a General Matrix to Hessenberg Form, Num. Math. 12,349-368 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/13, 339-358, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

The accuracy of ELMBAK can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real general matrices. In these paths, this subroutine is, in practice, numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

```
      SUBROUTINE ELMBAK(NM,LOW,IGH,A,INT,M,Z)
C
      INTEGER I,J,M,LA,MM,MP,NM,IGH,KP1,LOW,MP1
      REAL A(NM,IGH),Z(NM,M)
      REAL X
      INTEGER INT(IGH)
C
      IF (M .EQ. 0) GO TO 200
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C     ********** FOR MP=IGH-1 STEP -1 UNTIL LOW+1 DO -- **********
      DO 140 MM = KP1, LA
         MP = LOW + IGH - MM
         MP1 = MP + 1
C
         DO 110 I = MP1, IGH
            X = A(I,MP-1)
            IF (X .EQ. 0.0) GO TO 110
C
            DO 100 J = 1, M
  100       Z(I,J) = Z(I,J) + X * Z(MP,J)
C
  110    CONTINUE
C
         I = INT(MP)
         IF (I .EQ. MP) GO TO 140
C
         DO 130 J = 1, M
            X = Z(I,J)
            Z(I,J) = Z(MP,J)
            Z(MP,J) = X
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F273 ELMHES

A Fortran IV Subroutine to Reduce a Real General Matrix to Upper Hessenberg Form Using Elementary Transformations.

May, 1972

1. PURPOSE.

The Fortran IV subroutine ELMHES reduces a real general matrix to upper Hessenberg form using stabilized elementary similarity transformations. This reduced form is used by other subroutines to find the eigenvalues and/or eigenvectors of the original matrix. See section 2C for the specific routines.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  ELMHES(NM,N,LOW,IGH,A,INT)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional array A as specified in the DIMENSION statement for A in the calling program.

N is an integer input variable set equal to the order of the matrix A. N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F269 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

A is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, A contains the matrix of order N to be reduced to Hessenberg form. On output, A contains the upper Hessenberg matrix as well as the multipliers used in the reduction. See section 3 for the details.

INT is an integer output one-dimensional variable of dimension at least IGH identifying the rows and columns interchanged during the reduction. See section 3 for the details. Only components LOW+1 through IGH-1 are actually used by ELMHES.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues of the original matrix are desired, this subroutine should be followed by HQR (F286).

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by ELTRAN (F220) and HQR2 (F287).

If some of the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by HQR (F286), INVIT (F288), and ELMBAK (F274).

The subroutine ELMHES executes faster than its counterpart ORTHES (F275), which uses orthogonal similarity transformations. ORTHES is, however, more accurate in some cases. It is recommended that ELMHES be used in general.

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix A has the form

$$A = \begin{pmatrix} T & X & Y \\ 0 & B & Z \\ 0 & 0 & R \end{pmatrix}$$

where T and R are upper triangular matrices, B is a square matrix situated in rows and columns LOW through IGH, and X, Y, and Z are rectangular matrices of the appropriate dimensions. Then the subroutine ELMHES performs permutation and elementary similarity transformations to reduce B to Hessenberg form.

The Hessenberg reduction is performed in the following way. Starting with J=LOW, a permutation similarity transformation is performed to stabilize the succeeding elementary transformations in the J-th column of A. Next, each non-zero element below the subdiagonal in the J-th column of A is eliminated using elementary similarity transformations.

The permutation similarity transformation is determined by searching for the element of maximum modulus in the J-th column below the subdiagonal. This element is placed into the subdiagonal position by a row interchange and the corresponding column interchange is made to complete the permutation transformation. The index of the row interchanged with the (J+1)-th row is stored in INT(J+1).

The elementary transformation consists of elementary row operations on the J-th column of A to eliminate the non-zero elements below the subdiagonal and the corresponding elementary column operations to complete the similarity. The multipliers used in this elimination process are stored in place of the eliminated elements. These multipliers as well as the row interchange information in INT are used later in ELTRAN for the accumulation of the transformations and in ELMBAK for the back transformation of the eigenvectors.

The above steps are repeated on further columns of the transformed A until B is reduced to Hessenberg form; that is, repeated for J = LOW+1,LOW+2,...,IGH-2.

This subroutine is a translation of the Algol procedure ELMHES written and discussed in detail by Martin and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., Similarity Reduction of a General Matrix to Hessenberg Form, Num. Math. 12,349-368 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/13, 339-358, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

The accuracy of ELMHES can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real general matrices. In these paths, this subroutine is, in practice, numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

```
      SUBROUTINE ELMHES(NM,N,LOW,IGH,A,INT)
C
      INTEGER I,J,M,N,LA,NM,IGH,KP1,LOW,MM1,MP1
      REAL A(NM,N)
      REAL X,Y
      REAL ABS
      INTEGER INT(IGH)
C
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C
      DO 180 M = KP1, LA
         MM1 = M - 1
         X = 0.0
         I = M
C
         DO 100 J = M, IGH
            IF (ABS(A(J,MM1)) .LE. ABS(X)) GO TO 100
            X = A(J,MM1)
            I = J
  100    CONTINUE
C
         INT(M) = I
         IF (I .EQ. M) GO TO 130
C     ********** INTERCHANGE ROWS AND COLUMNS OF A **********
         DO 110 J = MM1, N
            Y = A(I,J)
            A(I,J) = A(M,J)
            A(M,J) = Y
  110    CONTINUE
C
         DO 120 J = 1, IGH
            Y = A(J,I)
            A(J,I) = A(J,M)
            A(J,M) = Y
  120    CONTINUE
C     ********** END INTERCHANGE **********
  130    IF (X .EQ. 0.0) GO TO 180
         MP1 = M + 1
C
         DO 160 I = MP1, IGH
            Y = A(I,MM1)
            IF (Y .EQ. 0.0) GO TO 160
            Y = Y / X
            A(I,MM1) = Y
C
            DO 140 J = M, N
  140       A(I,J) = A(I,J) - Y * A(M,J)
C
            DO 150 J = 1, IGH
  150       A(J,M) = A(J,M) + Y * A(J,I)
C
  160    CONTINUE
```

```
C
  180 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F220 ELTRAN

A Fortran IV Subroutine to Accumulate the Transformations in the Reduction of a Real General Matrix by ELMHES.

May, 1972

1. PURPOSE.

The Fortran IV subroutine ELTRAN accumulates the stabilized elementary similarity transformations used in the reduction of a real general matrix to upper Hessenberg form by ELMHES (F273).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE ELTRAN(NM,N,LOW,IGH,A,INT,Z)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

N is an integer input variable set equal to the order of the matrix A. N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F269 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

A is a working precision real input two-dimensional variable with row dimension NM and column dimension at least IGH. Its lower triangle below the subdiagonal contains the multipliers which were used in the reduction to the Hessenberg form. The remaining upper part of A is arbitrary. See section 3 of F273 for the details.

INT is an integer input one-dimensional variable of dimension at least IGH. INT identifies the rows and columns interchanged during the reduction by ELMHES. See section 3 of F273 for the details.

Z is a working precision real output two-dimensional variable with row dimension NM and column dimension at least N. It contains the transformation matrix produced in the reduction by ELMHES to the upper Hessenberg form.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine follows ELMHES (F273) and should be followed by HQR2 (F287). Otherwise, this subroutine will not ordinarily be used.

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix C (say) has been reduced to the upper Hessenberg form F stored in A by the similarity transformation

$$F = G^{-1} C G$$

where G is a product of the permutation and elementary matrices encoded in INT and in a lower triangle of A under F respectively. Then, ELTRAN accumulates G into the array Z.

This subroutine is a translation of the Algol procedure ELMTRANS written and discussed in detail by Peters and Wilkinson (1).

4. REFERENCES.

1) Peters, G. and Wilkinson, J.H., Eigenvectors of Real and Complex Matrices by LR and QR Triangularizations, Num. Math. 16,181-204 (1970). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/15, 372-395, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

ELTRAN introduces no rounding errors since it only transfers the multipliers used in the reduction process into the eigenvector matrix.

```
      SUBROUTINE ELTRAN(NM,N,LOW,IGH,A,INT,Z)
C
      INTEGER I,J,N,KL,MM,MP,NM,IGH,LOW,MP1
      REAL A(NM,IGH),Z(NM,N)
      INTEGER INT(IGH)
C
C     ********** INITIALIZE Z TO IDENTITY MATRIX **********
      DO 80 I = 1, N
C
         DO 60 J = 1, N
   60    Z(I,J) = 0.0
C
         Z(I,I) = 1.0
   80 CONTINUE
C
      KL = IGH - LOW - 1
      IF (KL .LT. 1) GO TO 200
C     ********** FOR MP=IGH-1 STEP -1 UNTIL LOW+1 DO -- **********
      DO 140 MM = 1, KL
         MP = IGH - MM
         MP1 = MP + 1
C
         DO 100 I = MP1, IGH
  100    Z(I,MP) = A(I,MP-1)
C
         I = INT(MP)
         IF (I .EQ. MP) GO TO 140
C
         DO 130 J = MP, IGH
            Z(MP,J) = Z(I,J)
            Z(I,J) = 0.0
  130    CONTINUE
C
         Z(I,MP) = 1.0
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F280-2 FIGI

A Fortran IV Subroutine to Transform a Certain Real
Non-Symmetric Tridiagonal Matrix to a
Symmetric Tridiagonal Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine FIGI transforms a certain real non-symmetric tridiagonal matrix to a symmetric tridiagonal matrix. The property of the non-symmetric matrix required for use of this subroutine is that the products of pairs of corresponding off-diagonal elements be all non-negative. The transformed matrix is used by other subroutines to find the eigenvalues and/or eigenvectors of the original matrix. See section 2C for the specific routines.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  FIGI(NM,N,T,D,E,E2,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array T as specified in the DIMENSION statement for T in the calling program.

N — is an integer input variable set equal to the order of the matrix T. N must be not greater than NM.

T is a working precision real input two-dimensional variable with row dimension NM and column dimension at least 3. T contains the non-symmetric tridiagonal matrix of order N in its first three columns. The subdiagonal elements are stored in the last N-1 positions of the first column. The diagonal elements are stored in the second column. The superdiagonal elements are stored in the first N-1 positions of the third column. Elements T(1,1) and T(N,3) are arbitrary.

D is a working precision real output one-dimensional variable of dimension at least N containing the diagonal elements of the tridiagonal symmetric matrix.

E is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the tridiagonal symmetric matrix. The element E(1) is not referenced.

E2 is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the squares of the subdiagonal elements of the tridiagonal symmetric matrix. The element E2(1) is not referenced. E2 need not be distinct from E (non-standard usage acceptable with at least those compilers included in the certification statement), in which case no squares are returned.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If the product of T(I,1) and T(I-1,3) is negative, FIGI terminates with IERR set to N+I. In this case, a symmetric matrix cannot be produced with this program.

If the product of T(I,1) and T(I-1,3) is zero but either T(I,1) or T(I-1,3) is not zero, FIGI sets IERR to -(3*N+I) and continues. If this occurs more than once, the last occurrence is recorded in IERR. See section 3.

If the products of all pairs of corresponding off-diagonal elements are non-negative, and zero only when both factors are zero, IERR is set to zero.

C. Applicability and Restrictions.

If all the eigenvalues of the original matrix are desired, this subroutine should be followed by IMTQL1 (F291).

If some of the eigenvalues of the original matrix are desired, this subroutine should be followed by BISECT (F294) or TRIDIB (F237).

If some of the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by TSTURM (F293), or by BISECT (F294) and TINVIT (F223), or by TRIDIB (F237) and TINVIT, or by IMTQLV (F234) and TINVIT, and then by BAKVEC (F281).

If all the eigenvalues and eigenvectors of the original matrix are desired, subroutine FIGI2 (F222) should be used rather than FIGI to perform the symmetrization, and should be followed by IMTQL2 (F292).

3. DISCUSSION OF METHOD AND ALGORITHM.

The transformation is performed in the following way. If the products of pairs of corresponding off-diagonal elements of T are all positive or if both elements are zero, a non-singular diagonal matrix V can be defined such that the similarity transformation

$$V^{-1} T V$$

produces a symmetric tridiagonal matrix with diagonal D and subdiagonal E as follows:

```
D(J) = T(J,2), J=1,2,3,...,N,
E(J) = SQRT(T(J,1) * T(J-1,3)), J=2,3,...,N.
```

The diagonal elements of the transformation matrix V are as follows:

```
V(1,1) = 1.0,
```

and for J=2,3,...,N

```
V(J,J) = V(J-1,J-1) * SQRT(T(J,1) / T(J-1,3))
                if  T(J,1) * T(J-1,3)  is positive,
```

or

```
V(J,J) = 1.0  if both  T(J,1)  and  T(J-1,3)  are zero.
```

If the product of a pair of corresponding off-diagonal elements is zero but only one of the elements is zero, then the symmetric tridiagonal matrix defined above has the same eigenvalues as T but its eigenvectors are not simply related to those of T. In this case, the back transformation subroutine BAKVEC (F281), if called, will set an error flag indicating that the computed eigenvectors are not valid.

This subroutine is an implementation of the algorithm discussed in detail by Wilkinson (1).

4. REFERENCES.

1) Wilkinson, J.H., The Algebraic Eigenvalue Problem, Oxford, Clarendon Press, 335-337 (1965).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The accuracy of FIGI can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of certain real non-symmetric tridiagonal matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE FIGI(NM,N,T,D,E,E2,IERR)
C
      INTEGER I,N,NM,IERR
      REAL T(NM,3),D(N),E(N),E2(N)
      REAL SQRT
C
      IERR = 0
C
      DO 100 I = 1, N
         IF (I .EQ. 1) GO TO 90
         E2(I) = T(I,1) * T(I-1,3)
         IF (E2(I)) 1000, 60, 80
   60    IF (T(I,1) .EQ. 0.0 .AND. T(I-1,3) .EQ. 0.0) GO TO 80
C     ********** SET ERROR -- PRODUCT OF SOME PAIR OF OFF-DIAGONAL
C                ELEMENTS IS ZERO WITH ONE MEMBER NON-ZERO **********
         IERR = -(3 * N + I)
   80    E(I) = SQRT(E2(I))
   90    D(I) = T(I,2)
  100 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- PRODUCT OF SOME PAIR OF OFF-DIAGONAL
C                ELEMENTS IS NEGATIVE **********
 1000 IERR = N + I
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F222 FIGI2

A Fortran IV Subroutine to Transform a Certain Real Non-Symmetric Tridiagonal Matrix to a Symmetric Tridiagonal Matrix Accumulating the Diagonal Transformations.

May, 1972

1. PURPOSE.

The Fortran IV subroutine FIGI2 transforms a certain real non-symmetric tridiagonal matrix to a symmetric tridiagonal matrix using and accumulating diagonal similarity transformations. The property of the non-symmetric matrix required for use of this subroutine is that the products of pairs of corresponding off-diagonal elements be all non-negative, and zero only when both factors are zero. The symmetrized matrix and the transformation matrix are used by subroutine IMTQL2 (F292) to find the eigenvalues and eigenvectors of the original matrix.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  FIGI2(NM,N,T,D,E,Z,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays T and Z as specified in the DIMENSION statements for T and Z in the calling program.

N — is an integer input variable set equal to the order of the matrix T. N must be not greater than NM.

T is a working precision real input two-dimensional variable with row dimension NM and column dimension at least 3. T contains the non-symmetric tridiagonal matrix of order N in its first three columns. The subdiagonal elements are stored in the last N-1 positions of the first column. The diagonal elements are stored in the second column. The superdiagonal elements are stored in the first N-1 positions of the third column. Elements T(1,1) and T(N,3) are arbitrary.

D is a working precision real output one-dimensional variable of dimension at least N containing the diagonal elements of the tridiagonal symmetric matrix.

E is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the tridiagonal symmetric matrix. The element E(1) is not referenced.

Z is a working precision real output two-dimensional variable with row dimension NM and column dimension at least N. It contains the diagonal transformation matrix produced in the symmetrization.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If the product of T(I,1) and T(I-1,3) is negative, FIGI2 terminates with IERR set to N+I. If the product of T(I,1) and T(I-1,3) is zero but either T(I,1) or T(I-1,3) is not zero, FIGI2 terminates with IERR set to 2*N+I. In these cases, there does not exist a symmetrizing similarity transformation, essential for the validity of the later eigenvector computation.

If the products of all pairs of corresponding off-diagonal elements are non-negative, and zero only when both factors are zero, IERR is set to zero.

C. Applicability and Restrictions.

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by IMTQL2 (F292).

If some other combination of eigenvalues and eigenvectors is desired, subroutine FIGI (F280) should be used rather than FIGI2 to perform the symmetrization.

3. DISCUSSION OF METHOD AND ALGORITHM.

The transformation is performed in the following way. If the products of pairs of corresponding off-diagonal elements of T are all positive or if both elements are zero, a non-singular diagonal matrix Z can be defined such that the similarity transformation

$$Z^{-1} T Z$$

produces a symmetric tridiagonal matrix with diagonal D and subdiagonal E as follows:

```
D(J) = T(J,2), J=1,2,3,...,N,
E(J) = SQRT(T(J,1) * T(J-1,3)), J=2,3,...,N.
```

The diagonal elements of the transformation matrix Z are as follows:

```
Z(1,1) = 1.0,
```

and for J=2,3,...,N

```
Z(J,J) = Z(J-1,J-1) * SQRT(T(J,1) / T(J-1,3))
                if  T(J,1) * T(J-1,3)  is positive,
```

or

```
Z(J,J) = 1.0  if both  T(J,1)  and  T(J-1,3)  are zero.
```

This subroutine is an implementation of the algorithm discussed in detail by Wilkinson (1).

4. REFERENCES.

1) Wilkinson, J.H., The Algebraic Eigenvalue Problem, Oxford, Clarendon Press, 335-337 (1965).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The accuracy of FIGI2 can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of certain real non-symmetric tridiagonal matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE FIGI2(NM,N,T,D,E,Z,IERR)
C
      INTEGER I,J,N,NM,IERR
      REAL T(NM,3),D(N),E(N),Z(NM,N)
      REAL H
      REAL SQRT
C
      IERR = 0
C
      DO 100 I = 1, N
C
         DO 50 J = 1, N
   50    Z(I,J) = 0.0
C
         IF (I .EQ. 1) GO TO 70
         H = T(I,1) * T(I-1,3)
         IF (H) 900, 60, 80
   60    IF (T(I,1) .NE. 0.0 .OR. T(I-1,3) .NE. 0.0) GO TO 1000
         E(I) = 0.0
   70    Z(I,I) = 1.0
         GO TO 90
   80    E(I) = SQRT(H)
         Z(I,I) = Z(I-1,I-1) * E(I) / T(I-1,3)
   90    D(I) = T(I,2)
  100 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- PRODUCT OF SOME PAIR OF OFF-DIAGONAL
C                ELEMENTS IS NEGATIVE **********
  900 IERR = N + I
      GO TO 1001
C     ********** SET ERROR -- PRODUCT OF SOME PAIR OF OFF-DIAGONAL
C                ELEMENTS IS ZERO WITH ONE MEMBER NON-ZERO **********
 1000 IERR = 2 * N + I
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F286-2 HQR

A Fortran IV Subroutine to Determine the Eigenvalues
of a Real Upper Hessenberg Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine HQR computes the eigenvalues of a real upper Hessenberg matrix using the QR method.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  HQR(NM,N,LOW,IGH,H,WR,WI,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional array H as specified in the DIMENSION statement for H in the calling program.

N is an integer input variable set equal to the order of the matrix H. N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F269 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

H is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, it contains the upper Hessenberg matrix. Note: HQR destroys this upper Hessenberg matrix as well as the two adjacent diagonals below it.

WR,WI are working precision real output one-dimensional variables of dimension at least N containing the real and imaginary parts, respectively, of the eigenvalues of the Hessenberg matrix. The eigenvalues are unordered except that complex conjugate pairs of eigenvalues appear consecutively with the eigenvalue having the positive imaginary part first.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, this subroutine terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the WR and WI arrays should be correct for indices IERR+1,IERR+2,...,N.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine some of the eigenvectors of a real Hessenberg matrix, HQR should be followed by INVIT (F288) to compute those eigenvectors. Note, however, that the upper Hessenberg matrix must be saved before HQR for later use by INVIT.

To determine the eigenvalues of a real general matrix, HQR must be preceded by ELMHES (F273) or ORTHES (F275) to produce a suitable upper Hessenberg matrix for HQR.

To determine some of the eigenvectors of a real general matrix, HQR should be preceded by ELMHES (F273) or ORTHES (F275) and followed by INVIT (F288) and ELMBAK (F274) or ORTBAK (F276). ELMBAK or ORTBAK back transforms the eigenvectors from INVIT into those of the real general matrix. Note, however, that the upper Hessenberg matrix and the two adjacent diagonals below it must be saved before HQR for the later use by INVIT and ELMBAK or ORTBAK, since HQR destroys this portion of H.

It is recommended in general that BALANC (F269) be used before ELMHES or ORTHES in which case BALBAK (F270) must be used after ELMBAK or ORTBAK if eigenvectors are computed.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the QR method. The essence of this method is a process whereby a sequence of upper Hessenberg matrices, unitarily similar to the original Hessenberg matrix, is formed which converges to a quasi-triangular matrix, that is, an upper Hessenberg matrix whose eigenvalues are the eigenvalues of 1x1 or 2x2 principal submatrices. The rate of convergence of this sequence is improved by shifting the origin at each iteration. The arithmetic throughout the process is kept real by combining two iterations into one, using two real origin shifts or a pair of complex conjugate origin shifts. Before each iteration, the last Hessenberg form is checked for a possible splitting into submatrices. If a splitting occurs, only the lower submatrix participates in the next iteration.

The origin shifts at each iteration are the eigenvalues of the lowest 2x2 principal minor. Whenever a lowest 1x1 or 2x2 principal submatrix finally splits from the rest of the matrix, the eigenvalues of this submatrix are taken to be eigenvalues of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1 or 2. The tolerances in the splitting tests are proportional to the relative machine precision.

Some of the eigenvalues may have been isolated on the diagonal by the subroutine BALANC (F269). This information is transmitted to HQR through the parameters LOW and IGH. As a result, HQR immediately extracts the eigenvalues in rows 1 to LOW-1 and IGH+1 to N, and so applies the QR procedure to the submatrix situated in rows and columns LOW through IGH.

This subroutine is a translation of the Algol procedure HQR written and discussed in detail by Martin, Peters, and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S., Peters, G., and Wilkinson, J.H., The QR Algorithm for Real Hessenberg Matrices, Num. Math. 14,219-231 (1970). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/14, 359-371, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

The subroutine HQR is numerically stable (1); that is, each computed eigenvalue is exact for a matrix close to the original upper Hessenberg matrix.

```
      SUBROUTINE HQR(NM,N,LOW,IGH,H,WR,WI,IERR)
C
      INTEGER I,J,K,L,M,N,EN,LL,MM,NA,NM,IGH,ITS,LOW,MP2,ENM2,IERR
      REAL H(NM,N),WR(N),WI(N)
      REAL P,Q,R,S,T,W,X,Y,ZZ,NORM,MACHEP
      REAL SQRT,ABS,SIGN
      INTEGER MINO
      LOGICAL NOTLAS
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      NORM = 0.0
      K = 1
C     ********** STORE ROOTS ISOLATED BY BALANC
C                AND COMPUTE MATRIX NORM **********
      DO 50 I = 1, N
C
         DO 40 J = K, N
   40    NORM = NORM + ABS(H(I,J))
C
         K = I
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 50
         WR(I) = H(I,I)
         WI(I) = 0.0
   50 CONTINUE
C
      EN = IGH
      T = 0.0
C     ********** SEARCH FOR NEXT EIGENVALUES **********
   60 IF (EN .LT. LOW) GO TO 1001
      ITS = 0
      NA = EN - 1
      ENM2 = NA - 1
C     ********** LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW DO -- **********
   70 DO 80 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 100
         S = ABS(H(L-1,L-1)) + ABS(H(L,L))
         IF (S .EQ. 0.0) S = NORM
         IF (ABS(H(L,L-1)) .LE. MACHEP * S) GO TO 100
   80 CONTINUE
C     ********** FORM SHIFT **********
  100 X = H(EN,EN)
      IF (L .EQ. EN) GO TO 270
      Y = H(NA,NA)
      W = H(EN,NA) * H(NA,EN)
      IF (L .EQ. NA) GO TO 280
      IF (ITS .EQ. 30) GO TO 1000
```

```
      IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 130
C     ********** FORM EXCEPTIONAL SHIFT **********
      T = T + X
C
      DO 120 I = LOW, EN
  120 H(I,I) = H(I,I) - X
C
      S = ABS(H(EN,NA)) + ABS(H(NA,ENM2))
      X = 0.75 * S
      Y = X
      W = -0.4375 * S * S
  130 ITS = ITS + 1
C     ********** LOOK FOR TWO CONSECUTIVE SMALL
C                SUB-DIAGONAL ELEMENTS.
C                FOR M=EN-2 STEP -1 UNTIL L DO -- **********
      DO 140 MM = L, ENM2
         M = ENM2 + L - MM
         ZZ = H(M,M)
         R = X - ZZ
         S = Y - ZZ
         P = (R * S - W) / H(M+1,M) + H(M,M+1)
         Q = H(M+1,M+1) - ZZ - R - S
         R = H(M+2,M+1)
         S = ABS(P) + ABS(Q) + ABS(R)
         P = P / S
         Q = Q / S
         R = R / S
         IF (M .EQ. L) GO TO 150
         IF (ABS(H(M,M-1)) * (ABS(Q) + ABS(R)) .LE. MACHEP * ABS(P)
     X    * (ABS(H(M-1,M-1)) + ABS(ZZ) + ABS(H(M+1,M+1)))) GO TO 150
  140 CONTINUE
C
  150 MP2 = M + 2
C
      DO 160 I = MP2, EN
         H(I,I-2) = 0.0
         IF (I .EQ. MP2) GO TO 160
         H(I,I-3) = 0.0
  160 CONTINUE
C     ********** DOUBLE QR STEP INVOLVING ROWS L TO EN AND
C                COLUMNS M TO EN **********
      DO 260 K = M, NA
         NOTLAS = K .NE. NA
         IF (K .EQ. M) GO TO 170
         P = H(K,K-1)
         Q = H(K+1,K-1)
         R = 0.0
         IF (NOTLAS) R = H(K+2,K-1)
         X = ABS(P) + ABS(Q) + ABS(R)
         IF (X .EQ. 0.0) GO TO 260
         P = P / X
         Q = Q / X
         R = R / X
```

```
  170     S = SIGN(SQRT(P*P+Q*Q+R*R),P)
          IF (K .EQ. M) GO TO 180
          H(K,K-1) = -S * X
          GO TO 190
  180     IF (L .NE. M) H(K,K-1) = -H(K,K-1)
  190     P = P + S
          X = P / S
          Y = Q / S
          ZZ = R / S
          Q = Q / P
          R = R / P
C       ********** ROW MODIFICATION **********
          DO 210 J = K, EN
             P = H(K,J) + Q * H(K+1,J)
             IF (.NOT. NOTLAS) GO TO 200
             P = P + R * H(K+2,J)
             H(K+2,J) = H(K+2,J) - P * ZZ
  200        H(K+1,J) = H(K+1,J) - P * Y
             H(K,J) = H(K,J) - P * X
  210     CONTINUE
C
          J = MIN0(EN,K+3)
C       ********** COLUMN MODIFICATION **********
          DO 230 I = L, J
             P = X * H(I,K) + Y * H(I,K+1)
             IF (.NOT. NOTLAS) GO TO 220
             P = P + ZZ * H(I,K+2)
             H(I,K+2) = H(I,K+2) - P * R
  220        H(I,K+1) = H(I,K+1) - P * Q
             H(I,K) = H(I,K) - P
  230     CONTINUE
C
  260 CONTINUE
C
          GO TO 70
C       ********** ONE ROOT FOUND **********
  270 WR(EN) = X + T
          WI(EN) = 0.0
          EN = NA
          GO TO 60
C       ********** TWO ROOTS FOUND **********
  280 P = (Y - X) / 2.0
          Q = P * P + W
          ZZ = SQRT(ABS(Q))
          X = X + T
          IF (Q .LT. 0.0) GO TO 320
C       ********** REAL PAIR **********
          ZZ = P + SIGN(ZZ,P)
          WR(NA) = X + ZZ
          WR(EN) = WR(NA)
          IF (ZZ .NE. 0.0) WR(EN) = X - W / ZZ
          WI(NA) = 0.0
          WI(EN) = 0.0
          GO TO 330
```

```
C     ********** COMPLEX PAIR **********
  320 WR(NA) = X + P
      WR(EN) = X + P
      WI(NA) = ZZ
      WI(EN) = -ZZ
  330 EN = ENM2
      GO TO 60
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = EN
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F287-2 HQR2

A Fortran IV Subroutine to Determine the Eigenvalues and Eigenvectors of a Real Upper Hessenberg Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine HQR2 computes the eigenvalues and eigenvectors of a real upper Hessenberg matrix using the QR method. The eigenvectors of a real general matrix can also be computed if ELMHES (F273) and ELTRAN (F220) or ORTHES (F275) and ORTRAN (F221) have been used to reduce this general matrix to Hessenberg form and to accumulate the transformations.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  HQR2(NM,N,LOW,IGH,H,WR,WI,Z,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays H and Z as specified in the DIMENSION statements for H and Z in the calling program.

N — is an integer input variable set equal to the order of the matrix H. N must be not greater than NM.

LOW,IGH — are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F269 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

H is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, it contains the upper Hessenberg matrix. Note: HQR2 destroys this upper Hessenberg matrix.

WR,WI are working precision real output one-dimensional variables of dimension at least N containing the real and imaginary parts, respectively, of the eigenvalues of the Hessenberg matrix. The eigenvalues are unordered except that complex conjugate pairs of eigenvalues appear consecutively with the eigenvalue having the positive imaginary part first.

Z is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. If the eigenvectors of the upper Hessenberg matrix are desired, then on input, Z contains the identity matrix of order N, and on output, contains the real and imaginary parts of the eigenvectors of this Hessenberg matrix. If the eigenvectors of a real general matrix are desired, then on input, Z contains the transformation matrix produced in ELTRAN or ORTRAN which reduced the general matrix to Hessenberg form, and on output, contains the real and imaginary parts of the eigenvectors of this real general matrix. If the J-th eigenvalue is real, the J-th column of Z contains its eigenvector. If the J-th eigenvalue is complex with positive imaginary part, the J-th and (J+1)-th columns of Z contain the real and imaginary parts of its eigenvector. The conjugate of this vector is the eigenvector for the conjugate eigenvalue. The eigenvectors are not normalized.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, this subroutine terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the WR and WI arrays should be correct for indices IERR+1,IERR+2,...,N, but no eigenvectors are computed.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine the eigenvalues and eigenvectors of a real general matrix, HQR2 should be preceded by ELMHES (F273) or ORTHES (F275) to provide a suitable upper Hessenberg matrix for HQR2 and by ELTRAN or ORTRAN to accumulate the transformations.

It is recommended in general that BALANC (F269) be used before ELMHES or ORTHES in which case BALBAK (F270) must be used after HQR2.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the QR method. The essence of this method is a process whereby a sequence of upper Hessenberg matrices, unitarily similar to the original Hessenberg matrix, is formed which converges to a quasi-triangular matrix, that is, an upper Hessenberg matrix whose eigenvalues are the eigenvalues of 1x1 or 2x2 principal submatrices. The rate of convergence of this sequence is improved by shifting the origin at each iteration. The arithmetic throughout the process is kept real by combining two iterations into one, using two real origin shifts or a pair of complex conjugate origin shifts. Before each iteration, the last Hessenberg form is checked for a possible splitting into submatrices. If a splitting occurs, only the lower submatrix participates in the next iteration. The similarity transformations used in each iteration are accumulated in the Z array.

The origin shifts at each iteration are the eigenvalues of the lowest 2x2 principal minor. Whenever a lowest 1x1 or 2x2 principal submatrix finally splits from the rest of the matrix, the eigenvalues of this submatrix are taken to be eigenvalues of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of

order 1 or 2. The tolerances in the splitting tests are proportional to the relative machine precision. The eigenvectors of this quasi-triangular matrix are determined by a back substitution process and then transformed to the eigenvectors of the original matrix, using the information in Z.

Some of the eigenvalues may have been isolated on the diagonal by the subroutine BALANC (F269). This information is transmitted to HQR2 through the parameters LOW and IGH. As a result, HQR2 immediately extracts the eigenvalues in rows 1 to LOW-1 and IGH+1 to N, and so applies the QR procedure to the submatrix situated in rows and columns LOW through IGH.

This subroutine is a translation of the Algol procedure HQR2 written and discussed in detail by Peters and Wilkinson (1).

4. REFERENCES.

1) Peters, G. and Wilkinson, J.H., Eigenvectors of Real and Complex Matrices by LR and QR Triangularizations, Num. Math. 16,181-204 (1970). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/15, 372-395, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

The subroutine HQR2 is numerically stable (1); that is, each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original upper Hessenberg matrix.

```
      SUBROUTINE HQR2(NM,N,LOW,IGH,H,WR,WI,Z,IERR)
C
      INTEGER I,J,K,L,M,N,EN,II,JJ,LL,MM,NA,NM,NN,
     X        IGH,ITS,LOW,MP2,ENM2,IERR
      REAL H(NM,N),WR(N),WI(N),Z(NM,N)
      REAL P,Q,R,S,T,W,X,Y,RA,SA,VI,VR,ZZ,NORM,MACHEP
      REAL SQRT,ABS,SIGN
      INTEGER MIN0
      LOGICAL NOTLAS
      COMPLEX Z3
      COMPLEX CMPLX
      REAL REAL,AIMAG
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      NORM = 0.0
      K = 1
C     ********** STORE ROOTS ISOLATED BY BALANC
C                AND COMPUTE MATRIX NORM **********
      DO 50 I = 1, N
C
         DO 40 J = K, N
   40    NORM = NORM + ABS(H(I,J))
C
         K = I
         IF (I .GE. LOW .AND. I .LE. IGH) GO TO 50
         WR(I) = H(I,I)
         WI(I) = 0.0
   50 CONTINUE
C
      EN = IGH
      T = 0.0
C     ********** SEARCH FOR NEXT EIGENVALUES **********
   60 IF (EN .LT. LOW) GO TO 340
      ITS = 0
      NA = EN - 1
      ENM2 = NA - 1
C     ********** LOOK FOR SINGLE SMALL SUB-DIAGONAL ELEMENT
C                FOR L=EN STEP -1 UNTIL LOW DO -- **********
   70 DO 80 LL = LOW, EN
         L = EN + LOW - LL
         IF (L .EQ. LOW) GO TO 100
         S = ABS(H(L-1,L-1)) + ABS(H(L,L))
         IF (S .EQ. 0.0) S = NORM
         IF (ABS(H(L,L-1)) .LE. MACHEP * S) GO TO 100
   80 CONTINUE
```

```
C     ********** FORM SHIFT **********
  100 X = H(EN,EN)
      IF (L .EQ. EN) GO TO 270
      Y = H(NA,NA)
      W = H(EN,NA) * H(NA,EN)
      IF (L .EQ. NA) GO TO 280
      IF (ITS .EQ. 30) GO TO 1000
      IF (ITS .NE. 10 .AND. ITS .NE. 20) GO TO 130
C     ********** FORM EXCEPTIONAL SHIFT **********
      T = T + X
C
      DO 120 I = LOW, EN
  120 H(I,I) = H(I,I) - X
C
      S = ABS(H(EN,NA)) + ABS(H(NA,ENM2))
      X = 0.75 * S
      Y = X
      W = -0.4375 * S * S
  130 ITS = ITS + 1
C     ********** LOOK FOR TWO CONSECUTIVE SMALL
C                SUB-DIAGONAL ELEMENTS.
C                FOR M=EN-2 STEP -1 UNTIL L DO -- **********
      DO 140 MM = L, ENM2
         M = ENM2 + L - MM
         ZZ = H(M,M)
         R = X - ZZ
         S = Y - ZZ
         P = (R * S - W) / H(M+1,M) + H(M,M+1)
         Q = H(M+1,M+1) - ZZ - R - S
         R = H(M+2,M+1)
         S = ABS(P) + ABS(Q) + ABS(R)
         P = P / S
         Q = Q / S
         R = R / S
         IF (M .EQ. L) GO TO 150
         IF (ABS(H(M,M-1)) * (ABS(Q) + ABS(R)) .LE. MACHEP * ABS(P)
     X    * (ABS(H(M-1,M-1)) + ABS(ZZ) + ABS(H(M+1,M+1)))) GO TO 150
  140 CONTINUE
C
  150 MP2 = M + 2
C
      DO 160 I = MP2, EN
         H(I,I-2) = 0.0
         IF (I .EQ. MP2) GO TO 160
         H(I,I-3) = 0.0
  160 CONTINUE
C     ********** DOUBLE QR STEP INVOLVING ROWS L TO EN AND
C                COLUMNS M TO EN **********
      DO 260 K = M, NA
         NOTLAS = K .NE. NA
         IF (K .EQ. M) GO TO 170
         P = H(K,K-1)
         Q = H(K+1,K-1)
         R = 0.0
```

```
            IF (NOTLAS) R = H(K+2,K-1)
            X = ABS(P) + ABS(Q) + ABS(R)
            IF (X .EQ. 0.0) GO TO 260
            P = P / X
            Q = Q / X
            R = R / X
  170       S = SIGN(SQRT(P*P+Q*Q+R*R),P)
            IF (K .EQ. M) GO TO 180
            H(K,K-1) = -S * X
            GO TO 190
  180       IF (L .NE. M) H(K,K-1) = -H(K,K-1)
  190       P = P + S
            X = P / S
            Y = Q / S
            ZZ = R / S
            Q = Q / P
            R = R / P
C         ********** ROW MODIFICATION **********
            DO 210 J = K, N
               P = H(K,J) + Q * H(K+1,J)
               IF (.NOT. NOTLAS) GO TO 200
               P = P + R * H(K+2,J)
               H(K+2,J) = H(K+2,J) - P * ZZ
  200          H(K+1,J) = H(K+1,J) - P * Y
               H(K,J) = H(K,J) - P * X
  210       CONTINUE
C
            J = MINO(EN,K+3)
C         ********** COLUMN MODIFICATION **********
            DO 230 I = 1, J
               P = X * H(I,K) + Y * H(I,K+1)
               IF (.NOT. NOTLAS) GO TO 220
               P = P + ZZ * H(I,K+2)
               H(I,K+2) = H(I,K+2) - P * R
  220          H(I,K+1) = H(I,K+1) - P * Q
               H(I,K) = H(I,K) - P
  230       CONTINUE
C         ********** ACCUMULATE TRANSFORMATIONS **********
            DO 250 I = LOW, IGH
               P = X * Z(I,K) + Y * Z(I,K+1)
               IF (.NOT. NOTLAS) GO TO 240
               P = P + ZZ * Z(I,K+2)
               Z(I,K+2) = Z(I,K+2) - P * R
  240          Z(I,K+1) = Z(I,K+1) - P * Q
               Z(I,K) = Z(I,K) - P
  250       CONTINUE
C
  260    CONTINUE
C
         GO TO 70
```

```
C     ********** ONE ROOT FOUND **********
  270 H(EN,EN) = X + T
      WR(EN) = H(EN,EN)
      WI(EN) = 0.0
      EN = NA
      GO TO 60
C     ********** TWO ROOTS FOUND **********
  280 P = (Y - X) / 2.0
      Q = P * P + W
      ZZ = SQRT(ABS(Q))
      H(EN,EN) = X + T
      X = H(EN,EN)
      H(NA,NA) = Y + T
      IF (Q .LT. 0.0) GO TO 320
C     ********** REAL PAIR **********
      ZZ = P + SIGN(ZZ,P)
      WR(NA) = X + ZZ
      WR(EN) = WR(NA)
      IF (ZZ .NE. 0.0) WR(EN) = X - W / ZZ
      WI(NA) = 0.0
      WI(EN) = 0.0
      X = H(EN,NA)
      S = ABS(X) + ABS(ZZ)
      P = X / S
      Q = ZZ / S
      R = SQRT(P*P+Q*Q)
      P = P / R
      Q = Q / R
C     ********** ROW MODIFICATION **********
      DO 290 J = NA, N
         ZZ = H(NA,J)
         H(NA,J) = Q * ZZ + P * H(EN,J)
         H(EN,J) = Q * H(EN,J) - P * ZZ
  290 CONTINUE
C     ********** COLUMN MODIFICATION **********
      DO 300 I = 1, EN
         ZZ = H(I,NA)
         H(I,NA) = Q * ZZ + P * H(I,EN)
         H(I,EN) = Q * H(I,EN) - P * ZZ
  300 CONTINUE
C     ********** ACCUMULATE TRANSFORMATIONS **********
      DO 310 I = LOW, IGH
         ZZ = Z(I,NA)
         Z(I,NA) = Q * ZZ + P * Z(I,EN)
         Z(I,EN) = Q * Z(I,EN) - P * ZZ
  310 CONTINUE
C
      GO TO 330
```

```
C     ********** COMPLEX PAIR **********
  320 WR(NA) = X + P
      WR(EN) = X + P
      WI(NA) = ZZ
      WI(EN) = -ZZ
  330 EN = ENM2
      GO TO 60
C     ********** ALL ROOTS FOUND.  BACKSUBSTITUTE TO FIND
C                VECTORS OF UPPER TRIANGULAR FORM **********
  340 IF (NORM .EQ. 0.0) GO TO 1001
C     ********** FOR EN=N STEP -1 UNTIL 1 DO -- **********
      DO 800 NN = 1, N
         EN = N + 1 - NN
         P = WR(EN)
         Q = WI(EN)
         NA = EN - 1
         IF (Q) 710, 600, 800
C     ********** REAL VECTOR **********
  600    M = EN
         H(EN,EN) = 1.0
         IF (NA .EQ. 0) GO TO 800
C     ********** FOR I=EN-1 STEP -1 UNTIL 1 DO -- **********
         DO 700 II = 1, NA
            I = EN - II
            W = H(I,I) - P
            R = H(I,EN)
            IF (M .GT. NA) GO TO 620
C
            DO 610 J = M, NA
  610       R = R + H(I,J) * H(J,EN)
C
  620       IF (WI(I) .GE. 0.0) GO TO 630
            ZZ = W
            S = R
            GO TO 700
  630       M = I
            IF (WI(I) .NE. 0.0) GO TO 640
            T = W
            IF (W .EQ. 0.0) T = MACHEP * NORM
            H(I,EN) = -R / T
            GO TO 700
C     ********** SOLVE REAL EQUATIONS **********
  640       X = H(I,I+1)
            Y = H(I+1,I)
            Q = (WR(I) - P) * (WR(I) - P) + WI(I) * WI(I)
            T = (X * S - ZZ * R) / Q
            H(I,EN) = T
            IF (ABS(X) .LE. ABS(ZZ)) GO TO 650
            H(I+1,EN) = (-R - W * T) / X
            GO TO 700
  650       H(I+1,EN) = (-S - Y * T) / ZZ
  700    CONTINUE
C     ********** END REAL VECTOR **********
         GO TO 800
```

```
C     ********** COMPLEX VECTOR **********
  710    M = NA
C     ********** LAST VECTOR COMPONENT CHOSEN IMAGINARY SO THAT
C                EIGENVECTOR MATRIX IS TRIANGULAR **********
         IF (ABS(H(EN,NA)) .LE. ABS(H(NA,EN))) GO TO 720
         H(NA,NA) = Q / H(EN,NA)
         H(NA,EN) = -(H(EN,EN) - P) / H(EN,NA)
         GO TO 730
  720    Z3 = CMPLX(0.0,-H(NA,EN)) / CMPLX(H(NA,NA)-P,Q)
         H(NA,NA) = REAL(Z3)
         H(NA,EN) = AIMAG(Z3)
  730    H(EN,NA) = 0.0
         H(EN,EN) = 1.0
         ENM2 = NA - 1
         IF (ENM2 .EQ. 0) GO TO 800
C     ********** FOR I=EN-2 STEP -1 UNTIL 1 DO -- **********
         DO 790 II = 1, ENM2
            I = NA - II
            W = H(I,I) - P
            RA = 0.0
            SA = H(I,EN)
C
            DO 760 J = M, NA
               RA = RA + H(I,J) * H(J,NA)
               SA = SA + H(I,J) * H(J,EN)
  760       CONTINUE
C
            IF (WI(I) .GE. 0.0) GO TO 770
            ZZ = W
            R = RA
            S = SA
            GO TO 790
  770       M = I
            IF (WI(I) .NE. 0.0) GO TO 780
            Z3 = CMPLX(-RA,-SA) / CMPLX(W,Q)
            H(I,NA) = REAL(Z3)
            H(I,EN) = AIMAG(Z3)
            GO TO 790
C     ********** SOLVE COMPLEX EQUATIONS **********
  780       X = H(I,I+1)
            Y = H(I+1,I)
            VR = (WR(I) - P) * (WR(I) - P) + WI(I) * WI(I) - Q * Q
            VI = (WR(I) - P) * 2.0 * Q
            IF (VR .EQ. 0.0 .AND. VI .EQ. 0.0) VR = MACHEP * NORM
     X       * (ABS(W) + ABS(Q) + ABS(X) + ABS(Y) + ABS(ZZ))
            Z3 = CMPLX(X*R-ZZ*RA+Q*SA,X*S-ZZ*SA-Q*RA) / CMPLX(VR,VI)
            H(I,NA) = REAL(Z3)
            H(I,EN) = AIMAG(Z3)
            IF (ABS(X) .LE. ABS(ZZ) + ABS(Q)) GO TO 785
            H(I+1,NA) = (-RA - W * H(I,NA) + Q * H(I,EN)) / X
            H(I+1,EN) = (-SA - W * H(I,EN) - Q * H(I,NA)) / X
            GO TO 790
```

```
  785          Z3 = CMPLX(-R-Y*H(I,NA),-S-Y*H(I,EN)) / CMPLX(ZZ,Q)
               H(I+1,NA) = REAL(Z3)
               H(I+1,EN) = AIMAG(Z3)
  790       CONTINUE
C        ********** END COMPLEX VECTOR **********
  800    CONTINUE
C        ********** END BACK SUBSTITUTION.
C                   VECTORS OF ISOLATED ROOTS **********
         DO 840 I = 1, N
            IF (I .GE. LOW .AND. I .LE. IGH) GO TO 840
C
            DO 820 J = I, N
  820       Z(I,J) = H(I,J)
C
  840    CONTINUE
C        ********** MULTIPLY BY TRANSFORMATION MATRIX TO GIVE
C                   VECTORS OF ORIGINAL FULL MATRIX.
C                   FOR J=N STEP -1 UNTIL LOW DO -- **********
         DO 880 JJ = LOW, N
            J = N + LOW - JJ
            M = MIN0(J,IGH)
C
            DO 880 I = LOW, IGH
               ZZ = 0.0
C
               DO 860 K = LOW, M
  860          ZZ = ZZ + Z(I,K) * H(K,J)
C
               Z(I,J) = ZZ
  880    CONTINUE
C
         GO TO 1001
C        ********** SET ERROR -- NO CONVERGENCE TO AN
C                   EIGENVALUE AFTER 30 ITERATIONS **********
 1000    IERR = EN
 1001    RETURN
         END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F285-2 HTRIBK

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Symmetric Tridiagonal Matrix Determined by HTRIDI.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine HTRIBK forms the eigenvectors of a complex Hermitian matrix from the eigenvectors of that real symmetric tridiagonal matrix determined by HTRIDI (F284).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  HTRIBK(NM,N,AR,AI,TAU,M,ZR,ZI)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays AR, AI, ZR, and ZI as specified in the DIMENSION statements for AR, AI, ZR, and ZI in the calling program.

N — is an integer input variable set equal to the order of the matrix. N must be not greater than NM.

AR,AI — are working precision real input two-dimensional variables with row dimension NM and column dimension at least N. The strict lower triangle of AR and the full lower triangle of AI contain some information about the unitary transformations used in the reduction to the tridiagonal form. The remaining upper parts of the matrices are arbitrary. See section 3 of F284 for the details.

TAU is a working precision real input two-dimensional variable with row dimension 2 and column dimension at least N. TAU contains the remaining information about the unitary transformations. See section 3 of F284 for the details.

M is an integer input variable set equal to the number of columns of Z = (ZR,ZI) to be back transformed.

ZR,ZI are working precision real two-dimensional variables with row dimension NM and column dimension at least M. On input, the first M columns of ZR contain the real eigenvectors to be back transformed and the contents of ZI are immaterial. On output, these M columns of ZR and ZI contain the real and imaginary parts, respectively, of the transformed eigenvectors. The transformed eigenvectors are orthonormal if the input eigenvectors are orthonormal.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine HTRIDI (F284).

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the Hermitian matrix C (say) has been reduced to the tridiagonal symmetric matrix F by the similarity transformation

$$F = V\ Q\ ^{*}\ CQV\ ^{*}$$

where Q is a product of the unitary Hermitian matrices encoded in the lower triangles of AR and AI, and V is a unitary diagonal matrix encoded in TAU. Then, given an array Z of column vectors, HTRIBK computes the matrix product QVZ. If the eigenvectors of F are columns of the array Z, then HTRIBK forms the eigenvectors of C in their place. Since QV is unitary, vector Euclidean norms are preserved. Note that the last component of each transformed eigenvector is real.

This subroutine is a complex analogue of the subroutine TRBAK1 (F279) which is a translation of the Algol procedure TRBAK1 written and discussed in detail by Martin, Reinsch, and Wilkinson (1). A similar Algol procedure REVERSE is discussed in detail by Mueller (2).

4. REFERENCES.

1) Martin, R.S., Reinsch, C., and Wilkinson, J.H., Householder's Tridiagonalization of a Symmetric Matrix, Num. Math. 11,181-195 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/2, 212-226, Springer-Verlag, 1971.)

2) Mueller, D.J., Householder's Method for Complex Matrices and Eigensystems of Hermitian Matrices, Num. Math. 8,72-92 (1966).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian matrices.

B. Accuracy.

The accuracy of HTRIBK can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of complex Hermitian matrices. In these paths, this subroutine is numerically stable (1,2). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE HTRIBK(NM,N,AR,AI,TAU,M,ZR,ZI)
C
      INTEGER I,J,K,L,M,N,NM
      REAL AR(NM,N),AI(NM,N),TAU(2,N),ZR(NM,M),ZI(NM,M)
      REAL H,S,SI
C
      IF (M .EQ. 0) GO TO 200
C     ********** TRANSFORM THE EIGENVECTORS OF THE REAL SYMMETRIC
C                TRIDIAGONAL MATRIX TO THOSE OF THE HERMITIAN
C                TRIDIAGONAL MATRIX. **********
      DO 50 K = 1, N
C
         DO 50 J = 1, M
            ZI(K,J) = -ZR(K,J) * TAU(2,K)
            ZR(K,J) = ZR(K,J) * TAU(1,K)
   50 CONTINUE
C
      IF (N .EQ. 1) GO TO 200
C     ********** RECOVER AND APPLY THE HOUSEHOLDER MATRICES **********
      DO 140 I = 2, N
         L = I - 1
         H = AI(I,I)
         IF (H .EQ. 0.0) GO TO 140
C
         DO 130 J = 1, M
            S = 0.0
            SI = 0.0
C
            DO 110 K = 1, L
               S = S + AR(I,K) * ZR(K,J) - AI(I,K) * ZI(K,J)
               SI = SI + AR(I,K) * ZI(K,J) + AI(I,K) * ZR(K,J)
  110       CONTINUE
C     ********** DOUBLE DIVISIONS AVOID POSSIBLE UNDERFLOW **********
            S = (S / H) / H
            SI = (SI / H) / H
C
            DO 120 K = 1, L
               ZR(K,J) = ZR(K,J) - S * AR(I,K) - SI * AI(I,K)
               ZI(K,J) = ZI(K,J) - SI * AR(I,K) + S * AI(I,K)
  120       CONTINUE
C
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F243 HTRIB3

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Symmetric Tridiagonal Matrix Determined by HTRID3.

July, 1975

1. PURPOSE.

The Fortran IV subroutine HTRIB3 forms the eigenvectors of a complex Hermitian matrix from the eigenvectors of that real symmetric tridiagonal matrix determined by HTRID3 (F242).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE HTRIB3(NM,N,A,TAU,M,ZR,ZI)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays A, ZR, and ZI as specified in the DIMENSION statements for A, ZR, and ZI in the calling program.

N is an integer input variable set equal to the order of the matrix. N must be not greater than NM.

A is a working precision real input two-dimensional variable with row dimension NM and column dimension at least N containing some information about the unitary transformations used in the reduction to the tridiagonal form. See section 3 of F242 for the details.

TAU is a working precision real input two-dimensional variable with row dimension 2 and column dimension at least N. TAU contains the remaining information about the unitary transformations. See section 3 of F242 for the details.

M is an integer input variable set equal to the number of columns of Z = (ZR,ZI) to be back transformed.

ZR,ZI are working precision real two-dimensional variables with row dimension NM and column dimension at least M. On input, the first M columns of ZR contain the real eigenvectors to be back transformed and the contents of ZI are immaterial. On output, these M columns of ZR and ZI contain the real and imaginary parts, respectively, of the transformed eigenvectors. The transformed eigenvectors are orthonormal if the input eigenvectors are orthonormal.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine HTRID3 (F242).

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the Hermitian matrix C (say) has been reduced to the tridiagonal symmetric matrix F by the similarity transformation

$$F = V\ Q\ ^{*}\ C\ ^{*}\ Q\ V$$

where Q is a product of the unitary Hermitian matrices encoded in A, and V is a unitary diagonal matrix encoded in TAU. Then, given an array Z of column vectors, HTRIB3 computes the matrix product QVZ. If the eigenvectors of F are columns of the array Z, then HTRIB3 forms the eigenvectors of C in their place. Since QV is unitary, vector Euclidean norms are preserved. Note that the last component of each transformed eigenvector is real.

This subroutine is a complex analogue of the subroutine TRBAK3 (F229) which is a translation of the Algol procedure TRBAK3 written and discussed in detail by Martin, Reinsch, and Wilkinson (1). A similar Algol procedure REVERSE is discussed in detail by Mueller (2).

4. REFERENCES.

1) Martin, R.S., Reinsch, C., and Wilkinson, J.H., Householder's Tridiagonalization of a Symmetric Matrix, Num. Math. 11,181-195 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/2, 212-226, Springer-Verlag, 1971.)

2) Mueller, D.J., Householder's Method for Complex Matrices and Eigensystems of Hermitian Matrices, Num. Math. 8,72-92 (1966).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian matrices.

B. Accuracy.

The accuracy of HTRIB3 can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of complex Hermitian matrices. In these paths, this subroutine is numerically stable (1,2). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE HTRIB3(NM,N,A,TAU,M,ZR,ZI)
C
      INTEGER I,J,K,L,M,N,NM
      REAL A(NM,N),TAU(2,N),ZR(NM,M),ZI(NM,M)
      REAL H,S,SI
C
      IF (M .EQ. 0) GO TO 200
C     ********** TRANSFORM THE EIGENVECTORS OF THE REAL SYMMETRIC
C                TRIDIAGONAL MATRIX TO THOSE OF THE HERMITIAN
C                TRIDIAGONAL MATRIX. **********
      DO 50 K = 1, N
C
         DO 50 J = 1, M
            ZI(K,J) = -ZR(K,J) * TAU(2,K)
            ZR(K,J) = ZR(K,J) * TAU(1,K)
   50 CONTINUE
C
      IF (N .EQ. 1) GO TO 200
C     ********** RECOVER AND APPLY THE HOUSEHOLDER MATRICES **********
      DO 140 I = 2, N
         L = I - 1
         H = A(I,I)
         IF (H .EQ. 0.0) GO TO 140
C
         DO 130 J = 1, M
            S = 0.0
            SI = 0.0
C
            DO 110 K = 1, L
               S = S + A(I,K) * ZR(K,J) - A(K,I) * ZI(K,J)
               SI = SI + A(I,K) * ZI(K,J) + A(K,I) * ZR(K,J)
  110       CONTINUE
C     ********** DOUBLE DIVISIONS AVOID POSSIBLE UNDERFLOW **********
            S = (S / H) / H
            SI = (SI / H) / H
C
            DO 120 K = 1, L
               ZR(K,J) = ZR(K,J) - S * A(I,K) - SI * A(K,I)
               ZI(K,J) = ZI(K,J) - SI * A(I,K) + S * A(K,I)
  120       CONTINUE
C
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F284-2 HTRIDI

A Fortran IV Subroutine to Reduce a Complex Hermitian Matrix to a Real Symmetric Tridiagonal Matrix Using Unitary Transformations.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine HTRIDI reduces a complex Hermitian matrix to a real symmetric tridiagonal matrix using unitary similarity transformations. This reduced form is used by other subroutines to find the eigenvalues and/or eigenvectors of the original matrix. See section 2C for the specific routines.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  HTRIDI(NM,N,AR,AI,D,E,E2,TAU)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays AR and AI as specified in the DIMENSION statements for AR and AI in the calling program.

N — is an integer input variable set equal to the order of the matrix A = (AR,AI). N must be not greater than NM.

AR,AI are working precision real two-dimensional variables with row dimension NM and column dimension at least N. On input, AR and AI contain the real and imaginary parts, respectively, of the Hermitian matrix of order N to be reduced to tridiagonal form. Only the full lower triangles of the matrices need be supplied. On output, the strict lower triangle of AR and the full lower triangle of AI contain some information about the unitary transformations used in the reduction. The full upper triangle of AR and the strict upper triangle of AI are unaltered. See section 3 for the details.

D is a working precision real output one-dimensional variable of dimension at least N containing the diagonal elements of the real symmetric tridiagonal matrix.

E is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the tridiagonal matrix. The element E(1) is set to zero.

E2 is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the squares of the subdiagonal elements of the tridiagonal matrix. The element E2(1) is set to zero. E2 need not be distinct from E (non-standard usage acceptable with at least those compilers included in the certification statement), in which case no squares are returned.

TAU is a working precision real output two-dimensional variable with row dimension 2 and column dimension at least N. TAU contains the remaining information about the unitary transformations. See section 3 for the details.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues of the original matrix are desired, this subroutine should be followed by TQL1 (F289), IMTQL1 (F291), or TQLRAT (F235).

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by TQL2 (F290) or IMTQL2 (F292), and then by HTRIBK (F285).

If some of the eigenvalues of the original matrix are desired, this subroutine should be followed by BISECT (F294) or TRIDIB (F237).

If some of the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by TSTURM (F293), or by BISECT (F294) and TINVIT (F223), or by TRIDIB (F237) and TINVIT, or by IMTQLV (F234) and TINVIT, and then by HTRIBK (F285).

3. DISCUSSION OF METHOD AND ALGORITHM.

The tridiagonal reduction is performed in the following way. Starting with J=N, the elements in the J-th row to the left of the diagonal of (AR,AI) are first scaled, to avoid possible underflow in the transformation that might result in severe departure from orthogonality. The sum of squared magnitudes SIGMA of these scaled elements is next formed. Then, a vector U and a scalar

$$H = U\,U^{*}/2$$

define an operator

$$P = I - UU^{*}/H$$

which is unitary and Hermitian and for which the similarity transformation PAP eliminates the elements in the J-th row of A to the left of the subdiagonal and the symmetrical elements in the J-th column.

The non-zero components of U are the elements of the J-th row to the left of the diagonal with the last of them augmented by the square root of SIGMA times the subdiagonal element divided by its magnitude. By storing the transformed subdiagonal element elsewhere and not overwriting the column elements eliminated in the transformation, full information about P is saved for later use in HTRIBK. The transformed subdiagonal element is then rendered real by a diagonal unitary transformation.

The above steps are repeated on further rows of the transformed A in reverse order until A is reduced to symmetric tridiagonal form; that is, repeated for J = N-1,N-2,...,3.

The product of the diagonal unitary transformations is accumulated into a unitary diagonal matrix whose diagonal elements are saved in TAU for later use in HTRIBK.

This subroutine is a complex analogue of the subroutine TRED1 (F277) which is a translation of the Algol procedure TRED1 written and discussed in detail by Martin, Reinsch, and Wilkinson (1). A similar Algol procedure HOUSEHOLDER HERMITIAN is discussed in detail by Mueller (2).

4. REFERENCES.

1) Martin, R.S., Reinsch, C., and Wilkinson, J.H., Householder's Tridiagonalization of a Symmetric Matrix, Num. Math. 11,181-195 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/2, 212-226, Springer-Verlag, 1971.)

2) Mueller, D.J., Householder's Method for Complex Matrices and Eigensystems of Hermitian Matrices, Num. Math. 8,72-92 (1966).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian matrices.

B. Accuracy.

The accuracy of HTRIDI can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of complex Hermitian matrices. In these paths, this subroutine is numerically stable (1,2). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE HTRIDI(NM,N,AR,AI,D,E,E2,TAU)
C
      INTEGER I,J,K,L,N,II,NM,JP1
      REAL AR(NM,N),AI(NM,N),D(N),E(N),E2(N),TAU(2,N)
      REAL F,G,H,FI,GI,HH,SI,SCALE
      REAL SQRT,CABS,ABS
      COMPLEX CMPLX
C
      TAU(1,N) = 1.0
      TAU(2,N) = 0.0
C
      DO 100 I = 1, N
  100 D(I) = AR(I,I)
C     ********** FOR I=N STEP -1 UNTIL 1 DO -- **********
      DO 300 II = 1, N
         I = N + 1 - II
         L = I - 1
         H = 0.0
         SCALE = 0.0
         IF (L .LT. 1) GO TO 130
C     ********** SCALE ROW (ALGOL TOL THEN NOT NEEDED) **********
         DO 120 K = 1, L
  120    SCALE = SCALE + ABS(AR(I,K)) + ABS(AI(I,K))
C
         IF (SCALE .NE. 0.0) GO TO 140
         TAU(1,L) = 1.0
         TAU(2,L) = 0.0
  130    E(I) = 0.0
         E2(I) = 0.0
         GO TO 290
C
  140    DO 150 K = 1, L
            AR(I,K) = AR(I,K) / SCALE
            AI(I,K) = AI(I,K) / SCALE
            H = H + AR(I,K) * AR(I,K) + AI(I,K) * AI(I,K)
  150    CONTINUE
C
         E2(I) = SCALE * SCALE * H
         G = SQRT(H)
         E(I) = SCALE * G
         F = CABS(CMPLX(AR(I,L),AI(I,L)))
C     ********** FORM NEXT DIAGONAL ELEMENT OF MATRIX T **********
         IF (F .EQ. 0.0) GO TO 160
         TAU(1,L) = (AI(I,L) * TAU(2,I) - AR(I,L) * TAU(1,I)) / F
         SI = (AR(I,L) * TAU(2,I) + AI(I,L) * TAU(1,I)) / F
         H = H + F * G
         G = 1.0 + G / F
         AR(I,L) = G * AR(I,L)
         AI(I,L) = G * AI(I,L)
         IF (L .EQ. 1) GO TO 270
         GO TO 170
  160    TAU(1,L) = -TAU(1,I)
         SI = TAU(2,I)
         AR(I,L) = G
```

```
  170     F = 0.0
C
          DO 240 J = 1, L
             G = 0.0
             GI = 0.0
C      ********** FORM ELEMENT OF A*U **********
             DO 180 K = 1, J
                G = G + AR(J,K) * AR(I,K) + AI(J,K) * AI(I,K)
                GI = GI - AR(J,K) * AI(I,K) + AI(J,K) * AR(I,K)
  180        CONTINUE
C
             JP1 = J + 1
             IF (L .LT. JP1) GO TO 220
C
             DO 200 K = JP1, L
                G = G + AR(K,J) * AR(I,K) - AI(K,J) * AI(I,K)
                GI = GI - AR(K,J) * AI(I,K) - AI(K,J) * AR(I,K)
  200        CONTINUE
C      ********** FORM ELEMENT OF P **********
  220        E(J) = G / H
             TAU(2,J) = GI / H
             F = F + E(J) * AR(I,J) - TAU(2,J) * AI(I,J)
  240     CONTINUE
C
          HH = F / (H + H)
C      ********** FORM REDUCED A **********
          DO 260 J = 1, L
             F = AR(I,J)
             G = E(J) - HH * F
             E(J) = G
             FI = -AI(I,J)
             GI = TAU(2,J) - HH * FI
             TAU(2,J) = -GI
C
             DO 260 K = 1, J
                AR(J,K) = AR(J,K) - F * E(K) - G * AR(I,K)
     X                            + FI * TAU(2,K) + GI * AI(I,K)
                AI(J,K) = AI(J,K) - F * TAU(2,K) - G * AI(I,K)
     X                            - FI * E(K) - GI * AR(I,K)
  260     CONTINUE
C
  270     DO 280 K = 1, L
             AR(I,K) = SCALE * AR(I,K)
             AI(I,K) = SCALE * AI(I,K)
  280     CONTINUE
C
          TAU(2,L) = -SI
  290     HH = D(I)
          D(I) = AR(I,I)
          AR(I,I) = HH
          AI(I,I) = SCALE * SQRT(H)
  300 CONTINUE
```

```
C
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F242 HTRID3

A Fortran IV Subroutine to Reduce a Complex Hermitian Matrix, Stored as a Single Square Array, to a Real Symmetric Tridiagonal Matrix Using Unitary Transformations.

July, 1975

1. PURPOSE.

The Fortran IV subroutine HTRID3 reduces a complex Hermitian matrix, stored as a single square array, to a real symmetric tridiagonal matrix using unitary similarity transformations. This reduced form is used by other subroutines to find the eigenvalues and/or eigenvectors of the original matrix. See section 2C for the specific routines.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  HTRID3(NM,N,A,D,E,E2,TAU)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional array A as specified in the DIMENSION statement for A in the calling program.

N is an integer input variable set equal to the order of the matrix. N must be not greater than NM.

A is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, A contains the lower triangle of the Hermitian matrix of order N to be reduced to tridiagonal form. If the real part of A

is denoted by AR and the imaginary part by AI, then the full lower triangle of AR should be stored in the full lower triangle of A and the strict lower triangle of AI should be stored in the transposed positions of the strict upper triangle of A. No storage is required for the zero diagonal elements of AI. For example when N=4, A should contain

```
( AR(1,1)   AI(2,1)   AI(3,1)   AI(4,1) )
( AR(2,1)   AR(2,2)   AI(3,2)   AI(4,2) )
( AR(3,1)   AR(3,2)   AR(3,3)   AI(4,3) )
( AR(4,1)   AR(4,2)   AR(4,3)   AR(4,4) ).
```

(If the upper triangle of AI is stored instead, the eigenvalues are not changed but the eigenvectors are conjugated.) On output, A contains some information about the unitary transformations used in the reduction. See section 3 for the details.

D is a working precision real output one-dimensional variable of dimension at least N containing the diagonal elements of the real symmetric tridiagonal matrix.

E is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the tridiagonal matrix. The element E(1) is set to zero.

E2 is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the squares of the subdiagonal elements of the tridiagonal matrix. The element E2(1) is set to zero. E2 need not be distinct from E (non-standard usage acceptable with at least those compilers included in the certification statement), in which case no squares are returned.

TAU is a working precision real output two-dimensional variable with row dimension 2 and column dimension at least N. TAU contains the remaining information about the unitary transformations. See section 3 for the details.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues of the original matrix are desired, this subroutine should be followed by TQL1 (F289), IMTQL1 (F291), or TQLRAT (F235).

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by TQL2 (F290) or IMTQL2 (F292), and then by HTRIB3 (F243).

If some of the eigenvalues of the original matrix are desired, this subroutine should be followed by BISECT (F294) or TRIDIB (F237).

If some of the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by TSTURM (F293), or by BISECT (F294) and TINVIT (F223), or by TRIDIB (F237) and TINVIT, or by IMTQLV (F234) and TINVIT, and then by HTRIB3 (F243).

3. DISCUSSION OF METHOD AND ALGORITHM.

The tridiagonal reduction is performed in the following way. Starting with J=N, the elements in the J-th row to the left of the diagonal of A are first scaled, to avoid possible underflow in the transformation that might result in severe departure from orthogonality. The sum of squared magnitudes SIGMA of these scaled elements is next formed. Then, a vector U and a scalar

$$H = U\,U^{*}/2$$

define an operator

$$P = I - UU^{*}/H$$

which is unitary and Hermitian and for which the similarity transformation PAP eliminates the elements in the J-th row of A to the left of the subdiagonal and the symmetrical elements in the J-th column.

The non-zero components of U are the elements of the J-th row to the left of the diagonal with the last of them augmented by the square root of SIGMA times the subdiagonal element divided by its magnitude. By storing the transformed subdiagonal element elsewhere and not overwriting the column elements eliminated in the transformation, full information about P is saved for later use in HTRIB3. The transformed subdiagonal element is then rendered real by a diagonal unitary transformation.

The above steps are repeated on further rows of the transformed A in reverse order until A is reduced to symmetric tridiagonal form; that is, repeated for J = N-1,N-2,...,3.

The product of the diagonal unitary transformations is accumulated into a unitary diagonal matrix whose diagonal elements are saved in TAU for later use in HTRIB3.

This subroutine is a complex analogue of the subroutine TRED3 (F228) which is a translation of the Algol procedure TRED3 written and discussed in detail by Martin, Reinsch, and Wilkinson (1). A similar Algol procedure HOUSEHOLDER HERMITIAN is discussed in detail by Mueller (2).

4. REFERENCES.

1) Martin, R.S., Reinsch, C., and Wilkinson, J.H., Householder's Tridiagonalization of a Symmetric Matrix, Num. Math. 11,181-195 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/2, 212-226, Springer-Verlag, 1971.)

2) Mueller, D.J., Householder's Method for Complex Matrices and Eigensystems of Hermitian Matrices, Num. Math. 8,72-92 (1966).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian matrices.

B. Accuracy.

The accuracy of HTRID3 can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of complex Hermitian matrices. In these paths, this subroutine is numerically stable (1,2). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE HTRID3(NM,N,A,D,E,E2,TAU)
C
      INTEGER I,J,K,L,N,II,NM,JM1,JP1
      REAL A(NM,N),D(N),E(N),E2(N),TAU(2,N)
      REAL F,G,H,FI,GI,HH,SI,SCALE
      REAL SQRT,CABS,ABS
      COMPLEX CMPLX
C
      TAU(1,N) = 1.0
      TAU(2,N) = 0.0
C     ********** FOR I=N STEP -1 UNTIL 1 DO -- **********
      DO 300 II = 1, N
         I = N + 1 - II
         L = I - 1
         H = 0.0
         SCALE = 0.0
         IF (L .LT. 1) GO TO 130
C     ********** SCALE ROW (ALGOL TOL THEN NOT NEEDED) **********
         DO 120 K = 1, L
  120    SCALE = SCALE + ABS(A(I,K)) + ABS(A(K,I))
C
         IF (SCALE .NE. 0.0) GO TO 140
         TAU(1,L) = 1.0
         TAU(2,L) = 0.0
  130    E(I) = 0.0
         E2(I) = 0.0
         GO TO 290
C
  140    DO 150 K = 1, L
            A(I,K) = A(I,K) / SCALE
            A(K,I) = A(K,I) / SCALE
            H = H + A(I,K) * A(I,K) + A(K,I) * A(K,I)
  150    CONTINUE
C
         E2(I) = SCALE * SCALE * H
         G = SQRT(H)
         E(I) = SCALE * G
         F = CABS(CMPLX(A(I,L),A(L,I)))
C     ********** FORM NEXT DIAGONAL ELEMENT OF MATRIX T **********
         IF (F .EQ. 0.0) GO TO 160
         TAU(1,L) = (A(L,I) * TAU(2,I) - A(I,L) * TAU(1,I)) / F
         SI = (A(I,L) * TAU(2,I) + A(L,I) * TAU(1,I)) / F
         H = H + F * G
         G = 1.0 + G / F
         A(I,L) = G * A(I,L)
         A(L,I) = G * A(L,I)
         IF (L .EQ. 1) GO TO 270
         GO TO 170
  160    TAU(1,L) = -TAU(1,I)
         SI = TAU(2,I)
         A(I,L) = G
  170    F = 0.0
```

```
C
         DO 240 J = 1, L
            G = 0.0
            GI = 0.0
            IF (J .EQ. 1) GO TO 190
            JM1 = J - 1
C     ********** FORM ELEMENT OF A*U **********
            DO 180 K = 1, JM1
               G = G + A(J,K) * A(I,K) + A(K,J) * A(K,I)
               GI = GI - A(J,K) * A(K,I) + A(K,J) * A(I,K)
  180       CONTINUE
C
  190       G = G + A(J,J) * A(I,J)
            GI = GI - A(J,J) * A(J,I)
            JP1 = J + 1
            IF (L .LT. JP1) GO TO 220
C
            DO 200 K = JP1, L
               G = G + A(K,J) * A(I,K) - A(J,K) * A(K,I)
               GI = GI - A(K,J) * A(K,I) - A(J,K) * A(I,K)
  200       CONTINUE
C     ********** FORM ELEMENT OF P **********
  220       E(J) = G / H
            TAU(2,J) = GI / H
            F = F + E(J) * A(I,J) - TAU(2,J) * A(J,I)
  240    CONTINUE
C
         HH = F / (H + H)
C     ********** FORM REDUCED A **********
         DO 260 J = 1, L
            F = A(I,J)
            G = E(J) - HH * F
            E(J) = G
            FI = -A(J,I)
            GI = TAU(2,J) - HH * FI
            TAU(2,J) = -GI
            A(J,J) = A(J,J) - 2.0 * (F * G + FI * GI)
            IF (J .EQ. 1) GO TO 260
            JM1 = J - 1
C
            DO 250 K = 1, JM1
               A(J,K) = A(J,K) - F * E(K) - G * A(I,K)
     X                         + FI * TAU(2,K) + GI * A(K,I)
               A(K,J) = A(K,J) - F * TAU(2,K) - G * A(K,I)
     X                         - FI * E(K) - GI * A(I,K)
  250       CONTINUE
C
  260    CONTINUE
C
  270    DO 280 K = 1, L
            A(I,K) = SCALE * A(I,K)
            A(K,I) = SCALE * A(K,I)
  280    CONTINUE
```

```
C
         TAU(2,L) = -SI
  290    D(I) = A(I,I)
         A(I,I) = SCALE * SQRT(H)
  300 CONTINUE
C
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F234 IMTQLV

A Fortran IV Subroutine to Determine the Eigenvalues of a Symmetric Tridiagonal Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine IMTQLV is a variant of IMTQL1 (F291) that determines the eigenvalues of a symmetric tridiagonal matrix using the implicit QL method, and associates with them their corresponding submatrix indices. This feature and the preservation of the input matrix make it possible to later determine the eigenvectors of the matrix.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  IMTQLV(N,D,E,E2,W,IND,IERR,RV1)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

N — is an integer input variable set equal to the order of the matrix.

D — is a working precision real input one-dimensional variable of dimension at least N containing the diagonal elements of the symmetric tridiagonal matrix.

E — is a working precision real input one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary.

E2 is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the squares of the subdiagonal elements of the symmetric tridiagonal matrix. E2(1) is arbitrary. On output, E2(1) is set to zero. If any of the elements in E are regarded as negligible, the corresponding elements of E2 are set to zero, indicative of the splitting of the matrix into a direct sum of submatrices.

W is a working precision real output one-dimensional variable of dimension at least N containing the eigenvalues of the symmetric tridiagonal matrix. The eigenvalues are in ascending order in W.

IND is an integer output one-dimensional variable of dimension at least N containing the submatrix indices associated with the corresponding eigenvalues in W. Eigenvalues belonging to the first submatrix have index 1, those belonging to the second submatrix have index 2, etc.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

RV1 is a working precision real temporary one-dimensional variable of dimension at least N used to copy the subdiagonal array E which would otherwise be destroyed.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, IMTQLV terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the W array should be correct for indices 1,2,...,IERR-1. These eigenvalues are ordered but are not necessarily the smallest IERR-1 eigenvalues.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

The subroutine TQL1 (F289) can also be used to compute the eigenvalues of a symmetric tridiagonal matrix. (No variant of TQL1 is available that associates submatrix indices with the eigenvalues.) Note, however, that TQL1 does not perform well on matrices whose successive row sums vary widely in magnitude and are not strictly increasing from the first to the last row. IMTQLV is not sensitive to such row sums and is therefore especially recommended for symmetric tridiagonal matrices whose structure is not known.

To determine the eigenvalues of a full symmetric matrix, IMTQLV should be preceded by TRED1 (F277) to provide a suitable symmetric tridiagonal matrix for IMTQLV.

To determine the eigenvalues of a complex Hermitian matrix, IMTQLV should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for IMTQLV.

Note, however, that although IMTQLV may perform better on symmetric tridiagonal matrices poorly structured in the above sense, TRED1 and HTRIDI may also perform poorly on such full symmetric and complex Hermitian matrices. Hence, any advantage in using IMTQLV over TQL1 may be overshadowed by a poor performance in TRED1 or HTRIDI.

The eigenvalues of certain non-symmetric tridiagonal matrices can be computed using the combination of FIGI (F280) and IMTQLV. See F280 for the description of this special class of matrices. For such matrices, IMTQLV should be preceded by FIGI to provide a suitable symmetric matrix for IMTQLV.

To determine eigenvectors associated with the computed eigenvalues, IMTQLV should be followed by TINVIT (F223) and the appropriate back transformation subroutine -- TRBAK1 (F279) after TRED1, HTRIBK (F285) after HTRIDI, or BAKVEC (F281) after FIGI.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the implicit QL method. The essence of this method is a process whereby a sequence of symmetric tridiagonal matrices, unitarily similar to the original symmetric tridiagonal matrix, is formed which converges to a diagonal matrix. The rate of convergence of this sequence is improved by implicitly shifting the origin

at each iteration. Before each iteration, the symmetric tridiagonal matrix is checked for a possible splitting into submatrices. If a splitting occurs, only the uppermost submatrix participates in the next iteration. The eigenvalues are ordered in ascending order as they are found.

The origin shift at each iteration is the eigenvalue of the current uppermost 2x2 principal minor closer to the first diagonal element of this minor. Whenever the uppermost 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision.

This subroutine is a modification of the subroutine IMTQL1 which is a translation of the Algol procedure IMTQL1 written and discussed in detail by Martin and Wilkinson (1) and modified by Dubrulle (2).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., The Implicit QL Algorithm, Num. Math. 12,377-383 (1968).

2) Dubrulle, A., A Short Note on the Implicit QL Algorithm, Num. Math. 15,450 (1970).
(Combined With (1) in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/4, 241-248, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine IMTQLV is numerically stable (1,2); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix.

```
      SUBROUTINE IMTQLV(N,D,E,E2,W,IND,IERR,RV1)
C
      INTEGER I,J,K,L,M,N,II,MML,TAG,IERR
      REAL D(N),E(N),E2(N),W(N),RV1(N)
      REAL B,C,F,G,P,R,S,MACHEP
      REAL SQRT,ABS,SIGN
      INTEGER IND(N)
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      K = 0
      TAG = 0
C
      DO 100 I = 1, N
         W(I) = D(I)
         IF (I .NE. 1) RV1(I-1) = E(I)
  100 CONTINUE
C
      E2(1) = 0.0
      RV1(N) = 0.0
C
      DO 290 L = 1, N
         J = 0
C     ********** LOOK FOR SMALL SUB-DIAGONAL ELEMENT **********
  105    DO 110 M = L, N
            IF (M .EQ. N) GO TO 120
            IF (ABS(RV1(M)) .LE. MACHEP * (ABS(W(M)) + ABS(W(M+1))))
     X         GO TO 120
C     ********** GUARD AGAINST UNDERFLOWED ELEMENT OF E2 **********
            IF (E2(M+1) .EQ. 0.0) GO TO 125
  110    CONTINUE
C
  120    IF (M .LE. K) GO TO 130
         IF (M .NE. N) E2(M+1) = 0.0
  125    K = M
         TAG = TAG + 1
  130    P = W(L)
         IF (M .EQ. L) GO TO 215
         IF (J .EQ. 30) GO TO 1000
         J = J + 1
C     ********** FORM SHIFT **********
         G = (W(L+1) - P) / (2.0 * RV1(L))
         R = SQRT(G*G+1.0)
         G = W(M) - P + RV1(L) / (G + SIGN(R,G))
         S = 1.0
         C = 1.0
         P = 0.0
         MML = M - L
```

```
C     ********** FOR I=M-1 STEP -1 UNTIL L DO -- **********
         DO 200 II = 1, MML
            I = M - II
            F = S * RV1(I)
            B = C * RV1(I)
            IF (ABS(F) .LT. ABS(G)) GO TO 150
            C = G / F
            R = SQRT(C*C+1.0)
            RV1(I+1) = F * R
            S = 1.0 / R
            C = C * S
            GO TO 160
  150       S = F / G
            R = SQRT(S*S+1.0)
            RV1(I+1) = G * R
            C = 1.0 / R
            S = S * C
  160       G = W(I+1) - P
            R = (W(I) - G) * S + 2.0 * C * B
            P = S * R
            W(I+1) = G + P
            G = C * R - B
  200    CONTINUE
C
         W(L) = W(L) - P
         RV1(L) = G
         RV1(M) = 0.0
         GO TO 105
C     ********** ORDER EIGENVALUES **********
  215    IF (L .EQ. 1) GO TO 250
C     ********** FOR I=L STEP -1 UNTIL 2 DO -- **********
         DO 230 II = 2, L
            I = L + 2 - II
            IF (P .GE. W(I-1)) GO TO 270
            W(I) = W(I-1)
            IND(I) = IND(I-1)
  230    CONTINUE
C
  250    I = 1
  270    W(I) = P
         IND(I) = TAG
  290 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = L
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F291 IMTQL1

A Fortran IV Subroutine to Determine the Eigenvalues of a Symmetric Tridiagonal Matrix.

May, 1972

1. PURPOSE.

The Fortran IV subroutine IMTQL1 determines the eigenvalues of a symmetric tridiagonal matrix using the implicit QL method.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE IMTQL1(N,D,E,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

N — is an integer input variable set equal to the order of the matrix.

D — is a working precision real one-dimensional variable of dimension at least N. On input, it contains the diagonal elements of the symmetric tridiagonal matrix. On output, it contains the eigenvalues of this matrix in ascending order.

E — is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary. Note that IMTQL1 destroys E.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, IMTQL1 terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the D array should be correct for indices 1,2,...,IERR-1. These eigenvalues are ordered but are not necessarily the smallest IERR-1 eigenvalues.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

The subroutine TQL1 (F289) can also be used to compute the eigenvalues of a symmetric tridiagonal matrix. Note, however, that TQL1 does not perform well on matrices whose successive row sums vary widely in magnitude and are not strictly increasing from the first to the last row. IMTQL1 is not sensitive to such row sums and is therefore especially recommended for symmetric tridiagonal matrices whose structure is not known.

To determine the eigenvalues of a full symmetric matrix, IMTQL1 should be preceded by TRED1 (F277) to provide a suitable symmetric tridiagonal matrix for IMTQL1.

To determine the eigenvalues of a complex Hermitian matrix, IMTQL1 should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for IMTQL1.

Note, however, that although IMTQL1 may perform better on symmetric tridiagonal matrices poorly structured in the above sense, TRED1 and HTRIDI may also perform poorly on such full symmetric and complex Hermitian matrices. Hence, any advantage in using IMTQL1 over TQL1 may be overshadowed by a poor performance in TRED1 or HTRIDI.

The eigenvalues of certain non-symmetric tridiagonal matrices can be computed using the combination of FIGI (F280) and IMTQL1. See F280 for the description of this special class of matrices. For such matrices, IMTQL1 should be preceded by FIGI to provide a suitable symmetric matrix for IMTQL1.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the implicit QL method. The essence of this method is a process whereby a sequence of symmetric tridiagonal matrices, unitarily similar to the original symmetric tridiagonal matrix, is formed which converges to a diagonal matrix. The rate of convergence of this sequence is improved by implicitly shifting the origin at each iteration. Before each iteration, the symmetric tridiagonal matrix is checked for a possible splitting into submatrices. If a splitting occurs, only the uppermost submatrix participates in the next iteration. The eigenvalues are ordered in ascending order as they are found.

The origin shift at each iteration is the eigenvalue of the current uppermost 2x2 principal minor closer to the first diagonal element of this minor. Whenever the uppermost 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision.

This subroutine is a translation of the Algol procedure IMTQL1 written and discussed in detail by Martin and Wilkinson (1) and modified by Dubrulle (2).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., The Implicit QL Algorithm, Num. Math. 12,377-383 (1968).

2) Dubrulle, A., A Short Note on the Implicit QL Algorithm, Num. Math. 15,450 (1970). (Combined With (1) in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/4, 241-248, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine IMTQL1 is numerically stable (1,2); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix.

```
      SUBROUTINE IMTQL1(N,D,E,IERR)
C
      INTEGER I,J,L,M,N,II,MML,IERR
      REAL D(N),E(N)
      REAL B,C,F,G,P,R,S,MACHEP
      REAL SQRT,ABS,SIGN
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      IF (N .EQ. 1) GO TO 1001
C
      DO 100 I = 2, N
  100 E(I-1) = E(I)
C
      E(N) = 0.0
C
      DO 290 L = 1, N
         J = 0
C     ********** LOOK FOR SMALL SUB-DIAGONAL ELEMENT **********
  105    DO 110 M = L, N
            IF (M .EQ. N) GO TO 120
            IF (ABS(E(M)) .LE. MACHEP * (ABS(D(M)) + ABS(D(M+1))))
     X         GO TO 120
  110    CONTINUE
C
  120    P = D(L)
         IF (M .EQ. L) GO TO 215
         IF (J .EQ. 30) GO TO 1000
         J = J + 1
C     ********** FORM SHIFT **********
         G = (D(L+1) - P) / (2.0 * E(L))
         R = SQRT(G*G+1.0)
         G = D(M) - P + E(L) / (G + SIGN(R,G))
         S = 1.0
         C = 1.0
         P = 0.0
         MML = M - L
C     ********** FOR I=M-1 STEP -1 UNTIL L DO -- **********
         DO 200 II = 1, MML
            I = M - II
            F = S * E(I)
            B = C * E(I)
            IF (ABS(F) .LT. ABS(G)) GO TO 150
            C = G / F
            R = SQRT(C*C+1.0)
            E(I+1) = F * R
            S = 1.0 / R
            C = C * S
            GO TO 160
```

```
  150        S = F / G
             R = SQRT(S*S+1.0)
             E(I+1) = G * R
             C = 1.0 / R
             S = S * C
  160        G = D(I+1) - P
             R = (D(I) - G) * S + 2.0 * C * B
             P = S * R
             D(I+1) = G + P
             G = C * R - B
  200     CONTINUE
C
          D(L) = D(L) - P
          E(L) = G
          E(M) = 0.0
          GO TO 105
C      ********** ORDER EIGENVALUES **********
  215     IF (L .EQ. 1) GO TO 250
C      ********** FOR I=L STEP -1 UNTIL 2 DO -- **********
          DO 230 II = 2, L
             I = L + 2 - II
             IF (P .GE. D(I-1)) GO TO 270
             D(I) = D(I-1)
  230     CONTINUE
C
  250     I = 1
  270     D(I) = P
  290  CONTINUE
C
       GO TO 1001
C      ********** SET ERROR -- NO CONVERGENCE TO AN
C                 EIGENVALUE AFTER 30 ITERATIONS **********
 1000  IERR = L
 1001  RETURN
       END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F292 IMTQL2

A Fortran IV Subroutine to Determine the Eigenvalues and Eigenvectors of a Symmetric Tridiagonal Matrix.

May, 1972

1. PURPOSE.

The Fortran IV subroutine IMTQL2 determines the eigenvalues and eigenvectors of a symmetric tridiagonal matrix. IMTQL2 uses the implicit QL method to compute the eigenvalues and accumulates the QL transformations to compute the eigenvectors. The eigenvectors of a full symmetric matrix can also be computed directly by IMTQL2, if TRED2 (F278) has been used to reduce this matrix to tridiagonal form.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  IMTQL2(NM,N,D,E,Z,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array Z as specified in the DIMENSION statement for Z in the calling program.

N — is an integer input variable set equal to the order of the matrix. N must be not greater than NM.

D — is a working precision real one-dimensional variable of dimension at least N. On input, it contains the diagonal elements of the symmetric tridiagonal matrix. On output, it contains the eigenvalues of this matrix in ascending order.

E — is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary. Note that IMTQL2 destroys E.

Z — is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. If the eigenvectors of the symmetric tridiagonal matrix are desired, then on input, Z contains the identity matrix of order N, and on output, contains the orthonormal eigenvectors of this tridiagonal matrix. If the eigenvectors of a full symmetric matrix are desired, then on input, Z contains the transformation matrix produced in TRED2 which reduced the full matrix to tridiagonal form, and on output, contains the orthonormal eigenvectors of this full symmetric matrix.

IERR — is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, IMTQL2 terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues and eigenvectors in the D and Z arrays should be correct for indices 1,2,...,IERR-1, but the eigenvalues are unordered.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

The subroutine TQL2 (F290) can also be used to compute the eigenvalues and eigenvectors of a symmetric tridiagonal matrix. Note, however, that TQL2 does not perform well on matrices whose successive row sums vary widely in magnitude and are not strictly increasing from the first to the last row. IMTQL2 is not sensitive to such row sums and is therefore especially recommended for symmetric tridiagonal matrices whose structure is not known.

To determine the eigenvalues and eigenvectors of a full symmetric matrix, IMTQL2 should be preceded by TRED2 (F278) to provide a suitable symmetric tridiagonal matrix for IMTQL2.

To determine the eigenvalues and eigenvectors of a complex Hermitian matrix, IMTQL2 should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for IMTQL2, and the input array Z to IMTQL2 should be initialized to the identity matrix. IMTQL2 should then be followed by HTRIBK (F285) to back transform the eigenvectors from IMTQL2 into those of the original matrix.

Note, however, that although IMTQL2 may perform better on symmetric tridiagonal matrices poorly structured in the above sense, TRED2 and HTRIDI may also perform poorly on such full symmetric and complex Hermitian matrices. Hence, any advantage in using IMTQL2 over TQL2 may be overshadowed by a poor performance in TRED2 or HTRIDI.

The eigenvalues and eigenvectors of certain non-symmetric tridiagonal matrices can be computed using the combination of FIGI2 (F222) and IMTQL2. See F222 for the description of this special class of matrices. For such matrices, IMTQL2 should be preceded by FIGI2 to provide a suitable symmetric matrix for IMTQL2.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the implicit QL method. The essence of this method is a process whereby a sequence of symmetric tridiagonal matrices, unitarily similar to the original symmetric tridiagonal matrix, is formed which converges to a diagonal matrix. The rate of convergence of this sequence is improved by implicitly shifting the origin at each iteration. Before each iteration, the symmetric tridiagonal matrix is checked for a possible splitting into submatrices. If a splitting occurs, only the uppermost submatrix participates in the next iteration. The similarity transformations used in each iteration are accumulated in the Z array, producing the orthonormal eigenvectors for the original matrix. Finally, the eigenvalues are ordered in ascending order and the eigenvectors are ordered consistently.

The origin shift at each iteration is the eigenvalue of the current uppermost 2x2 principal minor closer to the first diagonal element of this minor. Whenever the uppermost 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the

original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision.

This subroutine is a translation of the Algol procedure IMTQL2 written and discussed in detail by Martin and Wilkinson (1) and modified by Dubrulle (2).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., The Implicit QL Algorithm, Num. Math. 12,377-383 (1968).

2) Dubrulle, A., A Short Note on the Implicit QL Algorithm, Num. Math. 15,450 (1970). (Combined With (1) in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/4, 241-248, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine IMTQL2 is numerically stable (1,2); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE IMTQL2(NM,N,D,E,Z,IERR)
C
      INTEGER I,J,K,L,M,N,II,NM,MML,IERR
      REAL D(N),E(N),Z(NM,N)
      REAL B,C,F,G,P,R,S,MACHEP
      REAL SQRT,ABS,SIGN
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      IF (N .EQ. 1) GO TO 1001
C
      DO 100 I = 2, N
  100 E(I-1) = E(I)
C
      E(N) = 0.0
C
      DO 240 L = 1, N
         J = 0
C     ********** LOOK FOR SMALL SUB-DIAGONAL ELEMENT **********
  105    DO 110 M = L, N
            IF (M .EQ. N) GO TO 120
            IF (ABS(E(M)) .LE. MACHEP * (ABS(D(M)) + ABS(D(M+1))))
     X         GO TO 120
  110    CONTINUE
C
  120    P = D(L)
         IF (M .EQ. L) GO TO 240
         IF (J .EQ. 30) GO TO 1000
         J = J + 1
C     ********** FORM SHIFT **********
         G = (D(L+1) - P) / (2.0 * E(L))
         R = SQRT(G*G+1.0)
         G = D(M) - P + E(L) / (G + SIGN(R,G))
         S = 1.0
         C = 1.0
         P = 0.0
         MML = M - L
C     ********** FOR I=M-1 STEP -1 UNTIL L DO -- **********
         DO 200 II = 1, MML
            I = M - II
            F = S * E(I)
            B = C * E(I)
            IF (ABS(F) .LT. ABS(G)) GO TO 150
            C = G / F
            R = SQRT(C*C+1.0)
            E(I+1) = F * R
            S = 1.0 / R
            C = C * S
            GO TO 160
```

```
  150       S = F / G
            R = SQRT(S*S+1.0)
            E(I+1) = G * R
            C = 1.0 / R
            S = S * C
  160       G = D(I+1) - P
            R = (D(I) - G) * S + 2.0 * C * B
            P = S * R
            D(I+1) = G + P
            G = C * R - B
C     ********** FORM VECTOR **********
            DO 180 K = 1, N
               F = Z(K,I+1)
               Z(K,I+1) = S * Z(K,I) + C * F
               Z(K,I) = C * Z(K,I) - S * F
  180       CONTINUE
C
  200    CONTINUE
C
         D(L) = D(L) - P
         E(L) = G
         E(M) = 0.0
         GO TO 105
  240 CONTINUE
C     ********** ORDER EIGENVALUES AND EIGENVECTORS **********
      DO 300 II = 2, N
         I = II - 1
         K = I
         P = D(I)
C
         DO 260 J = II, N
            IF (D(J) .GE. P) GO TO 260
            K = J
            P = D(J)
  260    CONTINUE
C
         IF (K .EQ. I) GO TO 300
         D(K) = D(I)
         D(I) = P
C
         DO 280 J = 1, N
            P = Z(J,I)
            Z(J,I) = Z(J,K)
            Z(J,K) = P
  280    CONTINUE
C
  300 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = L
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F288-2 INVIT

A Fortran IV Subroutine to Determine Those Eigenvectors of a Real Upper Hessenberg Matrix Corresponding to Specified Eigenvalues.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine INVIT computes the eigenvectors of a real upper Hessenberg matrix corresponding to specified eigenvalues using inverse iteration.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  INVIT(NM,N,A,WR,WI,SELECT,MM,M,Z,
                                  IERR,RM1,RV1,RV2)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

N — is an integer input variable set equal to the order of the matrix A. N must be not greater than NM.

A — is a working precision real input two-dimensional variable with row dimension NM and column dimension at least N containing the upper Hessenberg matrix.

WR,WI are working precision real one-dimensional variables of dimension at least N. On input, they contain the real and imaginary parts, respectively, of the eigenvalues of the Hessenberg matrix. The eigenvalues need not be ordered, except that complex conjugate eigenvalues must appear consecutively and eigenvalues of any submatrix of the input Hessenberg matrix must have indices in WR,WI which lie between the boundary indices for that submatrix. These ordering constraints are satisfied by the output eigenvalues from HQR (F286). On output, the eigenvalues are unaltered, except that the real parts of close eigenvalues may be perturbed slightly in an attempt to obtain independent eigenvectors.

SELECT is a logical one-dimensional variable of dimension at least N. On input, the true elements flag those eigenvalues whose eigenvectors are desired. If both of a pair of complex conjugate eigenvalues are flagged, the second flag is set false and only the eigenvector corresponding to the first of these is computed.

MM is an integer input variable set equal to an upper bound for the number of columns required to store the real and imaginary parts of the requested eigenvectors. The eigenvector corresponding to a real (complex) eigenvalue requires one (two) column(s).

M is an integer output variable set equal to the number of columns actually used to store the eigenvectors.

Z is a working precision real output two-dimensional variable with row dimension NM and column dimension at least MM containing the eigenvectors corresponding to the flagged eigenvalues. The real and imaginary parts of complex eigenvectors are stored in consecutive columns with the real part first. The eigenvectors are packed into the columns of Z starting at the first column.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

RM1 is a working precision real temporary array variable of dimension at least N*N. This array holds the triangularized form of the upper Hessenberg matrix used in the inverse iteration process.

RV1,RV2 are working precision real temporary one-dimensional variables of dimension at least N. They hold the approximate eigenvectors during the inverse iteration process.

B. Error Conditions and Returns.

If more than MM columns of Z are necessary to store the eigenvectors corresponding to the flagged eigenvalues, INVIT terminates with IERR set to -(2*N+1). In this case, M is either MM or MM-1 and is the number of columns of Z containing eigenvectors already computed.

If none of the initial vectors for the inverse iteration process produces an acceptable approximation to an eigenvector, INVIT terminates the computation for that eigenvector and sets IERR to -K where K is the index of the associated eigenvalue. If this failure occurs for more than one eigenvalue, the last occurrence is recorded in IERR. The columns of Z corresponding to failures of the above sort are set to zero vectors.

If both of the above error conditions occur, INVIT terminates with IERR set to -(N+K) where K is the index of the last eigenvalue whose inverse iteration process failed. The columns of Z corresponding to failures in the inverse iteration process are set to zero vectors, and in this case, M is the number of columns of Z containing eigenvectors already computed.

If neither of the above error conditions occurs, INVIT sets IERR to zero.

C. Applicability and Restrictions.

To determine eigenvectors corresponding to some of the eigenvalues of a real general matrix, INVIT should be preceded by ELMHES (F273) or ORTHES (F275) and HQR (F286). ELMHES or ORTHES provides a suitable upper Hessenberg matrix for INVIT and HQR determines the eigenvalues of this Hessenberg matrix. INVIT should then be followed by ELMBAK (F274) or ORTBAK (F276) to back transform the eigenvectors from INVIT into those of the original matrix. Note: the upper Hessenberg matrix and the two adjacent subdiagonals must be saved before HQR, since HQR destroys this part of A.

It is recommended in general that BALANC (F269) be used before ELMHES or ORTHES in which case BALBAK (F270) must be used after ELMBAK or ORTBAK.

In this implementation, the arithmetic is real throughout except for complex division and complex absolute value.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvectors are determined by inverse iteration. In INVIT, this is a process whereby a vector X1 satisfying the matrix linear equation U*X1 = X0 is computed, where X0 is an initial vector and U is the upper triangular factor in the LU decomposition of the matrix B-W*I with partial pivoting. W is an approximate eigenvalue of a leading submatrix B of the Hessenberg matrix in A. The vector X1 is accepted as an eigenvector of B if the norm of X1 is sufficiently larger than the norm of X0. This eigenvector of B is transformed to an eigenvector of A by simply appending zero components to it.

The computations for the real and complex eigenvectors are separate. In the real case, the factor U in the LU decomposition is stored in the full lower triangle of the array RM1, and in the complex case, the real part of U is stored in the full lower triangle of RM1 and the imaginary part of U is stored in the upper triangle of RM1 above the superdiagonal augmented by the two columns of Z that will later store the complex eigenvector. (Internally, RM1 is treated as a two-dimensional NxN array.)

The acceptance criterion for X1 is a growth test of its norm. If MACHEP denotes the relative machine precision, then currently, X1 is rejected as an eigenvector if

SQRT(UK)*!!B!!*MACHEP*!!X1!! .LT. !!X0!!/10.0

where UK is the order of the submatrix B whose eigenvector is being computed and the norm !! !! is the infinity norm. At most UK orthogonal initial vectors X0 are tried to obtain the required growth. If no vector is accepted, the parameter IERR is set to indicate this failure and INVIT proceeds to compute the next eigenvector.

The real parts of close or identical eigenvalues whose eigenvectors are desired may be perturbed slightly in an attempt to obtain independent eigenvectors (if they exist). If required, such perturbations are positive and small multiples of MACHEP*!!B!!.

This subroutine is a translation of the Algol procedure INVIT written and discussed in detail by Peters and Wilkinson (1).

4. REFERENCES.

1) Peters, G. and Wilkinson, J.H., The Calculation of Specified Eigenvectors by Inverse Iteration, Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/18, 418-439, Springer-Verlag, 1971.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

The accuracy of INVIT can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real upper Hessenberg matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original upper Hessenberg matrix.

```
      SUBROUTINE INVIT(NM,N,A,WR,WI,SELECT,MM,M,Z,IERR,RM1,RV1,RV2)
C
      INTEGER I,J,K,L,M,N,S,II,IP,MM,MP,NM,NS,N1,UK,IP1,ITS,KM1,IERR
      REAL A(NM,N),WR(N),WI(N),Z(NM,MM),RM1(N,N),RV1(N),RV2(N)
      REAL T,W,X,Y,EPS3,NORM,NORMV,GROWTO,ILAMBD,MACHEP,RLAMBD,UKROOT
      REAL SQRT,CABS,ABS,FLOAT
      INTEGER IABS
      LOGICAL SELECT(N)
      COMPLEX Z3
      COMPLEX CMPLX
      REAL REAL,AIMAG
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      UK = 0
      S = 1
C     ********** IP = 0, REAL EIGENVALUE
C                      1, FIRST OF CONJUGATE COMPLEX PAIR
C                     -1, SECOND OF CONJUGATE COMPLEX PAIR **********
      IP = 0
      N1 = N - 1
C
      DO 980 K = 1, N
         IF (WI(K) .EQ. 0.0 .OR. IP .LT. 0) GO TO 100
         IP = 1
         IF (SELECT(K) .AND. SELECT(K+1)) SELECT(K+1) = .FALSE.
  100    IF (.NOT. SELECT(K)) GO TO 960
         IF (WI(K) .NE. 0.0) S = S + 1
         IF (S .GT. MM) GO TO 1000
         IF (UK .GE. K) GO TO 200
C     ********** CHECK FOR POSSIBLE SPLITTING **********
         DO 120 UK = K, N
            IF (UK .EQ. N) GO TO 140
            IF (A(UK+1,UK) .EQ. 0.0) GO TO 140
  120    CONTINUE
C     ********** COMPUTE INFINITY NORM OF LEADING UK BY UK
C                (HESSENBERG) MATRIX **********
  140    NORM = 0.0
         MP = 1
C
         DO 180 I = 1, UK
            X = 0.0
C
            DO 160 J = MP, UK
  160       X = X + ABS(A(I,J))
C
            IF (X .GT. NORM) NORM = X
            MP = I
  180    CONTINUE
```

```
C      ********** EPS3 REPLACES ZERO PIVOT IN DECOMPOSITION
C                 AND CLOSE ROOTS ARE MODIFIED BY EPS3 **********
         IF (NORM .EQ. 0.0) NORM = 1.0
         EPS3 = MACHEP * NORM
C      ********** GROWTO IS THE CRITERION FOR THE GROWTH **********
         UKROOT = SQRT(FLOAT(UK))
         GROWTO = 1.0E-1 / UKROOT
  200    RLAMBD = WR(K)
         ILAMBD = WI(K)
         IF (K .EQ. 1) GO TO 280
         KM1 = K - 1
         GO TO 240
C      ********** PERTURB EIGENVALUE IF IT IS CLOSE
C                 TO ANY PREVIOUS EIGENVALUE **********
  220    RLAMBD = RLAMBD + EPS3
C      ********** FOR I=K-1 STEP -1 UNTIL 1 DO -- **********
  240    DO 260 II = 1, KM1
            I = K - II
            IF (SELECT(I) .AND. ABS(WR(I)-RLAMBD) .LT. EPS3 .AND.
     X         ABS(WI(I)-ILAMBD) .LT. EPS3) GO TO 220
  260    CONTINUE
C
         WR(K) = RLAMBD
C      ********** PERTURB CONJUGATE EIGENVALUE TO MATCH **********
         IP1 = K + IP
         WR(IP1) = RLAMBD
C      ********** FORM UPPER HESSENBERG A-RLAMBD*I (TRANSPOSED)
C                 AND INITIAL REAL VECTOR **********
  280    MP = 1
C
         DO 320 I = 1, UK
C
            DO 300 J = MP, UK
  300       RM1(J,I) = A(I,J)
C
            RM1(I,I) = RM1(I,I) - RLAMBD
            MP = I
            RV1(I) = EPS3
  320    CONTINUE
C
         ITS = 0
         IF (ILAMBD .NE. 0.0) GO TO 520
C      ********** REAL EIGENVALUE.
C                 TRIANGULAR DECOMPOSITION WITH INTERCHANGES,
C                 REPLACING ZERO PIVOTS BY EPS3 **********
         IF (UK .EQ. 1) GO TO 420
C
         DO 400 I = 2, UK
            MP = I - 1
            IF (ABS(RM1(MP,I)) .LE. ABS(RM1(MP,MP))) GO TO 360
```

```
C
            DO 340 J = MP, UK
               Y = RM1(J,I)
               RM1(J,I) = RM1(J,MP)
               RM1(J,MP) = Y
  340       CONTINUE
C
  360       IF (RM1(MP,MP) .EQ. 0.0) RM1(MP,MP) = EPS3
            X = RM1(MP,I) / RM1(MP,MP)
            IF (X .EQ. 0.0) GO TO 400
C
            DO 380 J = I, UK
  380       RM1(J,I) = RM1(J,I) - X * RM1(J,MP)
C
  400    CONTINUE
C
  420    IF (RM1(UK,UK) .EQ. 0.0) RM1(UK,UK) = EPS3
C     ********** BACK SUBSTITUTION FOR REAL VECTOR
C                   FOR I=UK STEP -1 UNTIL 1 DO -- **********
  440    DO 500 II = 1, UK
            I = UK + 1 - II
            Y = RV1(I)
            IF (I .EQ. UK) GO TO 480
            IP1 = I + 1
C
            DO 460 J = IP1, UK
  460       Y = Y - RM1(J,I) * RV1(J)
C
  480       RV1(I) = Y / RM1(I,I)
  500    CONTINUE
C
         GO TO 740
C     ********** COMPLEX EIGENVALUE.
C                   TRIANGULAR DECOMPOSITION WITH INTERCHANGES,
C                   REPLACING ZERO PIVOTS BY EPS3.  STORE IMAGINARY
C                   PARTS IN UPPER TRIANGLE STARTING AT (1,3) **********
  520    NS = N - S
         Z(1,S-1) = -ILAMBD
         Z(1,S) = 0.0
         IF (N .EQ. 2) GO TO 550
         RM1(1,3) = -ILAMBD
         Z(1,S-1) = 0.0
         IF (N .EQ. 3) GO TO 550
C
         DO 540 I = 4, N
  540    RM1(1,I) = 0.0
C
  550    DO 640 I = 2, UK
            MP = I - 1
            W = RM1(MP,I)
            IF (I .LT. N) T = RM1(MP,I+1)
            IF (I .EQ. N) T = Z(MP,S-1)
            X = RM1(MP,MP) * RM1(MP,MP) + T * T
            IF (W * W .LE. X) GO TO 580
```

```
            X = RM1(MP,MP) / W
            Y = T / W
            RM1(MP,MP) = W
            IF (I .LT. N) RM1(MP,I+1) = 0.0
            IF (I .EQ. N) Z(MP,S-1) = 0.0
C
            DO 560 J = I, UK
               W = RM1(J,I)
               RM1(J,I) = RM1(J,MP) - X * W
               RM1(J,MP) = W
               IF (J .LT. N1) GO TO 555
               L = J - NS
               Z(I,L) = Z(MP,L) - Y * W
               Z(MP,L) = 0.0
               GO TO 560
  555          RM1(I,J+2) = RM1(MP,J+2) - Y * W
               RM1(MP,J+2) = 0.0
  560       CONTINUE
C
            RM1(I,I) = RM1(I,I) - Y * ILAMBD
            IF (I .LT. N1) GO TO 570
            L = I - NS
            Z(MP,L) = -ILAMBD
            Z(I,L) = Z(I,L) + X * ILAMBD
            GO TO 640
  570       RM1(MP,I+2) = -ILAMBD
            RM1(I,I+2) = RM1(I,I+2) + X * ILAMBD
            GO TO 640
  580       IF (X .NE. 0.0) GO TO 600
            RM1(MP,MP) = EPS3
            IF (I .LT. N) RM1(MP,I+1) = 0.0
            IF (I .EQ. N) Z(MP,S-1) = 0.0
            T = 0.0
            X = EPS3 * EPS3
  600       W = W / X
            X = RM1(MP,MP) * W
            Y = -T * W
C
            DO 620 J = I, UK
               IF (J .LT. N1) GO TO 610
               L = J - NS
               T = Z(MP,L)
               Z(I,L) = -X * T - Y * RM1(J,MP)
               GO TO 615
  610          T = RM1(MP,J+2)
               RM1(I,J+2) = -X * T - Y * RM1(J,MP)
  615          RM1(J,I) = RM1(J,I) - X * RM1(J,MP) + Y * T
  620       CONTINUE
```

```
C
            IF (I .LT. N1) GO TO 630
            L = I - NS
            Z(I,L) = Z(I,L) - ILAMBD
            GO TO 640
  630       RM1(I,I+2) = RM1(I,I+2) - ILAMBD
  640    CONTINUE
C
         IF (UK .LT. N1) GO TO 650
         L = UK - NS
         T = Z(UK,L)
         GO TO 655
  650    T = RM1(UK,UK+2)
  655    IF (RM1(UK,UK) .EQ. 0.0 .AND. T .EQ. 0.0) RM1(UK,UK) = EPS3
C     ********** BACK SUBSTITUTION FOR COMPLEX VECTOR
C                FOR I=UK STEP -1 UNTIL 1 DO -- **********
  660    DO 720 II = 1, UK
            I = UK + 1 - II
            X = RV1(I)
            Y = 0.0
            IF (I .EQ. UK) GO TO 700
            IP1 = I + 1
C
            DO 680 J = IP1, UK
               IF (J .LT. N1) GO TO 670
               L = J - NS
               T = Z(I,L)
               GO TO 675
  670          T = RM1(I,J+2)
  675          X = X - RM1(J,I) * RV1(J) + T * RV2(J)
               Y = Y - RM1(J,I) * RV2(J) - T * RV1(J)
  680       CONTINUE
C
  700       IF (I .LT. N1) GO TO 710
            L = I - NS
            T = Z(I,L)
            GO TO 715
  710       T = RM1(I,I+2)
  715       Z3 = CMPLX(X,Y) / CMPLX(RM1(I,I),T)
            RV1(I) = REAL(Z3)
            RV2(I) = AIMAG(Z3)
  720    CONTINUE
C     ********** ACCEPTANCE TEST FOR REAL OR COMPLEX
C                EIGENVECTOR AND NORMALIZATION **********
  740    ITS = ITS + 1
         NORM = 0.0
         NORMV = 0.0
C
         DO 780 I = 1, UK
            IF (ILAMBD .EQ. 0.0) X = ABS(RV1(I))
            IF (ILAMBD .NE. 0.0) X = CABS(CMPLX(RV1(I),RV2(I)))
            IF (NORMV .GE. X) GO TO 760
            NORMV = X
            J = I
```

```
  760        NORM = NORM + X
  780     CONTINUE
C
          IF (NORM .LT. GROWTO) GO TO 840
C      ********** ACCEPT VECTOR **********
          X = RV1(J)
          IF (ILAMBD .EQ. 0.0) X = 1.0 / X
          IF (ILAMBD .NE. 0.0) Y = RV2(J)
C
          DO 820 I = 1, UK
             IF (ILAMBD .NE. 0.0) GO TO 800
             Z(I,S) = RV1(I) * X
             GO TO 820
  800        Z3 = CMPLX(RV1(I),RV2(I)) / CMPLX(X,Y)
             Z(I,S-1) = REAL(Z3)
             Z(I,S) = AIMAG(Z3)
  820     CONTINUE
C
          IF (UK .EQ. N) GO TO 940
          J = UK + 1
          GO TO 900
C      ********** IN-LINE PROCEDURE FOR CHOOSING
C                 A NEW STARTING VECTOR **********
  840     IF (ITS .GE. UK) GO TO 880
          X = UKROOT
          Y = EPS3 / (X + 1.0)
          RV1(1) = EPS3
C
          DO 860 I = 2, UK
  860     RV1(I) = Y
C
          J = UK - ITS + 1
          RV1(J) = RV1(J) - EPS3 * X
          IF (ILAMBD .EQ. 0.0) GO TO 440
          GO TO 660
C      ********** SET ERROR -- UNACCEPTED EIGENVECTOR **********
  880     J = 1
          IERR = -K
C      ********** SET REMAINING VECTOR COMPONENTS TO ZERO **********
  900     DO 920 I = J, N
             Z(I,S) = 0.0
             IF (ILAMBD .NE. 0.0) Z(I,S-1) = 0.0
  920     CONTINUE
C
  940     S = S + 1
  960     IF (IP .EQ. (-1)) IP = 0
          IF (IP .EQ. 1) IP = -1
  980  CONTINUE
C
       GO TO 1001
```

```
C     ********** SET ERROR -- UNDERESTIMATE OF EIGENVECTOR
C                SPACE REQUIRED **********
 1000 IF (IERR .NE. 0) IERR = IERR - N
      IF (IERR .EQ. 0) IERR = -(2 * N + 1)
 1001 M = S - 1 - IABS(IP)
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F276-2 ORTBAK

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Upper Hessenberg Matrix Determined by ORTHES.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine ORTBAK forms the eigenvectors of a real general matrix from the eigenvectors of that upper Hessenberg matrix determined by ORTHES (F275).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE ORTBAK(NM,LOW,IGH,A,ORT,M,Z)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F269 for the details. If the matrix is not balanced, set LOW to 1 and IGH to the order of the matrix.

A is a working precision real input two-dimensional variable with row dimension NM and column dimension at least IGH. The strict lower triangle of A contains some information about the orthogonal

transformations used in the reduction to the Hessenberg form. The remaining upper part of the matrix is arbitrary. See section 3 of F275 for the details.

ORT is a working precision real one-dimensional variable of dimension at least IGH. On input, ORT contains further information about the orthogonal transformations used in the reduction by ORTHES. See section 3 of F275 for the details. ORT is used for temporary storage within ORTBAK and is not restored.

M is an integer input variable set equal to the number of columns of Z to be back transformed.

Z is a working precision real two-dimensional variable with row dimension NM and column dimension at least M. On input, the first M columns of Z contain the real and imaginary parts of the eigenvectors to be back transformed. On output, these M columns of Z contain the real and imaginary parts of the transformed eigenvectors.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine ORTHES (F275).

The real and imaginary parts of an eigenvector need not be stored in consecutive columns of Z.

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix C (say) has been reduced to the upper Hessenberg form F stored in A by the similarity transformation

$$F = Q^T C Q$$

where Q is a product of the orthogonal matrices encoded in ORT and in a lower triangle of A under F. Then, given an array Z of column vectors, ORTBAK computes the matrix product QZ. If the real and imaginary parts of the eigenvectors of F are columns of the array Z, then ORTBAK forms the eigenvectors of C in their place.

This subroutine is a translation of the Algol procedure ORTBAK written and discussed in detail by Martin and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., Similarity Reduction of a General Matrix to Hessenberg Form, Num. Math. 12,349-368 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/13, 339-358, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

The accuracy of ORTBAK can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real general matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

```
      SUBROUTINE ORTBAK(NM,LOW,IGH,A,ORT,M,Z)
C
      INTEGER I,J,M,LA,MM,MP,NM,IGH,KP1,LOW,MP1
      REAL A(NM,IGH),ORT(IGH),Z(NM,M)
      REAL G
C
      IF (M .EQ. 0) GO TO 200
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C     ********** FOR MP=IGH-1 STEP -1 UNTIL LOW+1 DO -- **********
      DO 140 MM = KP1, LA
         MP = LOW + IGH - MM
         IF (A(MP,MP-1) .EQ. 0.0) GO TO 140
         MP1 = MP + 1
C
         DO 100 I = MP1, IGH
  100    ORT(I) = A(I,MP-1)
C
         DO 130 J = 1, M
            G = 0.0
C
            DO 110 I = MP, IGH
  110       G = G + ORT(I) * Z(I,J)
C     ********** DIVISOR BELOW IS NEGATIVE OF H FORMED IN ORTHES.
C                DOUBLE DIVISION AVOIDS POSSIBLE UNDERFLOW **********
            G = (G / ORT(MP)) / A(MP,MP-1)
C
            DO 120 I = MP, IGH
  120       Z(I,J) = Z(I,J) + G * ORT(I)
C
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F275 ORTHES

A Fortran IV Subroutine to Reduce a Real General Matrix to Upper Hessenberg Form Using Orthogonal Transformations.

May, 1972

1. PURPOSE.

The Fortran IV subroutine ORTHES reduces a real general matrix to upper Hessenberg form using orthogonal similarity transformations. This reduced form is used by other subroutines to find the eigenvalues and/or eigenvectors of the original matrix. See section 2C for the specific routines.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  ORTHES(NM,N,LOW,IGH,A,ORT)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM is an integer input variable set equal to the row dimension of the two-dimensional array A as specified in the DIMENSION statement for A in the calling program.

N is an integer input variable set equal to the order of the matrix A. N must be not greater than NM.

LOW,IGH are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F269 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

A is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, A contains the matrix of order N to be reduced to Hessenberg form. On output, A contains the upper Hessenberg matrix as well as some information about the orthogonal transformations used in the reduction. See section 3 for the details.

ORT is a working precision real output one-dimensional variable of dimension at least IGH containing the remaining information about the orthogonal transformations. See section 3 for the details. Only components LOW+1 through IGH are actually used by ORTHES.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues of the original matrix are desired, this subroutine should be followed by HQR (F286).

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by ORTRAN (F221) and HQR2 (F287).

If some of the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by HQR (F286), INVIT (F288), and ORTBAK (F276).

The subroutine ORTHES executes slower than its counterpart ELMHES (F273), which uses elementary similarity transformations. ORTHES is, however, more accurate in some cases. It is recommended that ELMHES be used in general.

If the matrix has elements of widely varying magnitudes, the larger ones should be in the top left-hand corner.

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix A has the form

$$A = \begin{pmatrix} T & X & Y \\ 0 & B & Z \\ 0 & 0 & R \end{pmatrix}$$

where T and R are upper triangular matrices, B is a square matrix situated in rows and columns LOW through IGH, and X, Y, and Z are rectangular matrices of the appropriate dimensions. Then the subroutine ORTHES performs orthogonal similarity transformations to reduce B to Hessenberg form.

The Hessenberg reduction is performed in the following way. Starting with J=LOW, the elements in the J-th column below the diagonal are first scaled, to avoid possible underflow in the transformation that might result in severe departure from orthogonality. The sum of squares SIGMA of these scaled elements is next formed. Then, a vector U and a scalar

$$H = U^T U/2$$

define an operator

$$P = I - UU^T/H$$

which is orthogonal and symmetric and for which the similarity transformation PAP eliminates the elements in the J-th column of A below the subdiagonal.

The non-zero components of U are the elements of the J-th column below the diagonal with the first of them augmented by the square root of SIGMA prefixed by the sign of the subdiagonal element. By saving this component in the array ORT and not overwriting the column elements eliminated in the transformation, full information on P is saved for later use in ORTRAN and ORTBAK.

The transformation replaces the subdiagonal element with the square root of SIGMA prefixed by sign opposite to that of the replaced element.

The above steps are repeated on further columns of the transformed A until B is reduced to Hessenberg form; that is, repeated for J = LOW+1,LOW+2,...,IGH-2.

This subroutine is a translation of the Algol procedure ORTHES written and discussed in detail by Martin and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S. and Wilkinson, J.H., Similarity Reduction of a General Matrix to Hessenberg Form, Num. Math. 12,349-368 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/13, 339-358, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

The accuracy of ORTHES can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real general matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

```
      SUBROUTINE ORTHES(NM,N,LOW,IGH,A,ORT)
C
      INTEGER I,J,M,N,II,JJ,LA,MP,NM,IGH,KP1,LOW
      REAL A(NM,N),ORT(IGH)
      REAL F,G,H,SCALE
      REAL SQRT,ABS,SIGN
C
      LA = IGH - 1
      KP1 = LOW + 1
      IF (LA .LT. KP1) GO TO 200
C
      DO 180 M = KP1, LA
         H = 0.0
         ORT(M) = 0.0
         SCALE = 0.0
C     ********** SCALE COLUMN (ALGOL TOL THEN NOT NEEDED) **********
         DO 90 I = M, IGH
   90    SCALE = SCALE + ABS(A(I,M-1))
C
         IF (SCALE .EQ. 0.0) GO TO 180
         MP = M + IGH
C     ********** FOR I=IGH STEP -1 UNTIL M DO -- **********
         DO 100 II = M, IGH
            I = MP - II
            ORT(I) = A(I,M-1) / SCALE
            H = H + ORT(I) * ORT(I)
  100    CONTINUE
C
         G = -SIGN(SQRT(H),ORT(M))
         H = H - ORT(M) * G
         ORT(M) = ORT(M) - G
C     ********** FORM (I-(U*UT)/H) * A **********
         DO 130 J = M, N
            F = 0.0
C     ********** FOR I=IGH STEP -1 UNTIL M DO -- **********
            DO 110 II = M, IGH
               I = MP - II
               F = F + ORT(I) * A(I,J)
  110       CONTINUE
C
            F = F / H
C
            DO 120 I = M, IGH
  120       A(I,J) = A(I,J) - F * ORT(I)
C
  130    CONTINUE
C     ********** FORM (I-(U*UT)/H)*A*(I-(U*UT)/H) **********
         DO 160 I = 1, IGH
            F = 0.0
C     ********** FOR J=IGH STEP -1 UNTIL M DO -- **********
            DO 140 JJ = M, IGH
               J = MP - JJ
               F = F + ORT(J) * A(I,J)
  140       CONTINUE
```

```
C
            F = F / H
C
            DO 150 J = M, IGH
  150       A(I,J) = A(I,J) - F * ORT(J)
C
  160     CONTINUE
C
          ORT(M) = SCALE * ORT(M)
          A(M,M-1) = SCALE * G
  180 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F221-2 ORTRAN

A Fortran IV Subroutine to Accumulate the Transformations in the Reduction of a Real General Matrix by ORTHES.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine ORTRAN accumulates the orthogonal similarity transformations used in the reduction of a real general matrix to upper Hessenberg form by ORTHES (F275).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  ORTRAN(NM,N,LOW,IGH,A,ORT,Z)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

N — is an integer input variable set equal to the order of the matrix A. N must be not greater than NM.

LOW,IGH — are integer input variables indicating the boundary indices for the balanced matrix. See section 3 of F269 for the details. If the matrix is not balanced, set LOW to 1 and IGH to N.

A is a working precision real input two-dimensional variable with row dimension NM and column dimension at least IGH. The strict lower triangle of A contains some information about the orthogonal transformations used in the reduction to the Hessenberg form. The remaining upper part of the matrix is arbitrary. See section 3 of F275 for the details.

ORT is a working precision real one-dimensional variable of dimension at least IGH. On input, ORT contains further information about the orthogonal transformations used in the reduction by ORTHES. See section 3 of F275 for the details. ORT is used for temporary storage within ORTRAN and is not restored.

Z is a working precision real output two-dimensional variable with row dimension NM and column dimension at least N. It contains the transformation matrix produced in the reduction by ORTHES to the upper Hessenberg form.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine follows ORTHES (F275) and should be followed by HQR2 (F287). Otherwise, this subroutine will not ordinarily be used.

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the matrix C (say) has been reduced to the upper Hessenberg form F stored in A by the similarity transformation

$$F = Q^T C Q$$

where Q is a product of the orthogonal matrices encoded in ORT and in a lower triangle of A under F. Then, ORTRAN accumulates Q into the array Z.

This subroutine is a translation of the Algol procedure ORTRANS written and discussed in detail by Peters and Wilkinson (1).

4. REFERENCES.

1) Peters, G. and Wilkinson, J.H., Eigenvectors of Real and Complex Matrices by LR and QR Triangularizations, Num. Math. 16,181-204 (1970). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/15, 372-395, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

The accuracy of ORTRAN can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real general matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

```
      SUBROUTINE ORTRAN(NM,N,LOW,IGH,A,ORT,Z)
C
      INTEGER I,J,N,KL,MM,MP,NM,IGH,LOW,MP1
      REAL A(NM,IGH),ORT(IGH),Z(NM,N)
      REAL G
C
C     ********** INITIALIZE Z TO IDENTITY MATRIX **********
      DO 80 I = 1, N
C
         DO 60 J = 1, N
   60    Z(I,J) = 0.0
C
         Z(I,I) = 1.0
   80 CONTINUE
C
      KL = IGH - LOW - 1
      IF (KL .LT. 1) GO TO 200
C     ********** FOR MP=IGH-1 STEP -1 UNTIL LOW+1 DO -- **********
      DO 140 MM = 1, KL
         MP = IGH - MM
         IF (A(MP,MP-1) .EQ. 0.0) GO TO 140
         MP1 = MP + 1
C
         DO 100 I = MP1, IGH
  100    ORT(I) = A(I,MP-1)
C
         DO 130 J = MP, IGH
            G = 0.0
C
            DO 110 I = MP, IGH
  110       G = G + ORT(I) * Z(I,J)
C     ********** DIVISOR BELOW IS NEGATIVE OF H FORMED IN ORTHES.
C                DOUBLE DIVISION AVOIDS POSSIBLE UNDERFLOW **********
            G = (G / ORT(MP)) / A(MP,MP-1)
C
            DO 120 I = MP, IGH
  120       Z(I,J) = Z(I,J) + G * ORT(I)
C
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F298-2 RATQR

A Fortran IV Subroutine to Determine Some Extreme Eigenvalues of a Symmetric Tridiagonal Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine RATQR determines the algebraically smallest or largest eigenvalues of a symmetric tridiagonal matrix using the rational QR method with Newton corrections.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  RATQR(N,EPS1,D,E,E2,
                  M,W,IND,BD,TYPE,IDEF,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

N — is an integer input variable set equal to the order of the matrix.

EPS1 — is a working precision real variable. On input, EPS1 specifies a tolerance for the theoretical absolute error of the computed eigenvalues. If MACHEP denotes the relative machine precision, then if the input EPS1 is less than MACHEP times the magnitude of any of the computed eigenvalues, it is reset to a default value described in section 3. The theoretical absolute error in the K-th eigenvalue is usually not greater than K times the output EPS1.

D is a working precision real input one-dimensional variable of dimension at least N containing the diagonal elements of the symmetric tridiagonal matrix.

E is a working precision real input one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary.

E2 is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the squares of the subdiagonal elements of the symmetric tridiagonal matrix. E2(1) is arbitrary. On output, E2(1) is set to 0.0 if the smallest eigenvalues are found and to 2.0 if the largest eigenvalues are found. If any of the elements in E are regarded as negligible, the corresponding elements of E2 are set to zero, and so the matrix splits into a direct sum of submatrices.

M is an integer input variable set equal to the number of extreme eigenvalues desired.

W is a working precision real output one-dimensional variable of dimension at least N containing the M extreme eigenvalues of the symmetric tridiagonal matrix. If the smallest eigenvalues have been found, they are in ascending order in W. If the largest eigenvalues have been found, they are in descending order in W. W need not be distinct from D.

IND is an integer output one-dimensional variable of dimension at least N containing the submatrix indices associated with the corresponding M eigenvalues in W. Eigenvalues belonging to the first submatrix have index 1, those belonging to the second submatrix have index 2, etc.

BD is a working precision real output one-dimensional variable of dimension at least N. BD(K) is a tighter bound than K*EPS1 for the theoretical error of the K-th eigenvalue W(K), K = 1,2,...,M. BD need not be distinct from E2.

TYPE is a logical input variable set true if the smallest eigenvalues are to be found and set false if the largest eigenvalues are to be found.

IDEF is an integer input variable set to 1 if the symmetric tridiagonal matrix is known to be positive definite, set to -1 if the matrix is known to be negative definite, and set to 0 otherwise.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If IDEF is set to 1 and TYPE is set true when the input matrix is not positive definite, or if IDEF is set to -1 and TYPE is set false when the matrix is not negative definite, RATQR terminates with IERR set to 6*N+1. No eigenvalues are computed. Note: this error return may also be made if M has been inadvertently specified greater than N.

If successive iterates to the K-th eigenvalue are not strictly monotone increasing, RATQR terminates the calculation for that eigenvalue with IERR set to 5*N+K. The sum of all the shifts up to this point is taken as the eigenvalue and the program proceeds to the next eigenvalue calculation. If this failure occurs for more than one eigenvalue, the last occurrence is recorded in IERR.

If neither of the above error conditions occurs, IERR is set to zero.

C. Applicability and Restrictions.

To determine extreme eigenvalues of a full symmetric matrix, RATQR should be preceded by TRED1 (F277) to provide a suitable symmetric tridiagonal matrix for RATQR.

To determine extreme eigenvalues of a complex Hermitian matrix, RATQR should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for RATQR.

To determine extreme eigenvalues of certain non-symmetric tridiagonal matrices, RATQR should be preceded by FIGI (F280) to provide a suitable symmetric tridiagonal matrix for RATQR. See F280 for a description of this special class of matrices.

To determine eigenvectors associated with the computed eigenvalues, RATQR should be followed by TINVIT (F223) and the appropriate back transformation subroutine -- TRBAK1 (F279) after TRED1, HTRIBK (F285) after HTRIDI, or BAKVEC (F281) after FIGI.

The subroutine TRIDIB (F237) is generally faster and more accurate than RATQR if the eigenvalues are clustered.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the rational QR method with Newton corrections. The essence of this method is a process whereby a sequence of symmetric tridiagonal matrices, unitarily similar to the original symmetric tridiagonal matrix with shifted origin, is formed which converges to a matrix with zero diagonal elements. The origin shifts using the Newton step improve the convergence rate of the sequence.

The algorithm applies to the determination of the algebraically smallest eigenvalues. D is first copied into W and IDEF into variable JDEF. If TYPE is false, RATQR then reverses the signs of the elements of W and negates JDEF.

Next, the subdiagonal elements are tested for negligibility. If an element is considered negligible, its square is set to zero, and so the matrix splits into a direct sum of submatrices. Then if JDEF is not 1 or if JDEF is 1 and the Gerschgorin lower bound for the smallest eigenvalue is positive, the origin is first shifted so that the Gerschgorin lower bound for the translated matrix is zero.

Starting from the new origin, two QR decompositions are performed providing with additional calculations the Newton step by which the origin is again shifted. This process is repeated until a diagonal element of one of the intermediate matrices becomes negligible. Then the sum of the origin shifts is taken as an eigenvalue, the matrix is deflated by deleting the row and column with the negligible diagonal element (the location of this row serves to identify the submatrix to which the eigenvalue belongs), and the complete process is repeated on the deflated matrix.

If at any stage MACHEP times the absolute sum of the origin shifts exceeds EPS1, EPS1 is reset to this quantity. A diagonal element is considered negligible if it is no greater than the current EPS1. At each iteration, negligible squared subdiagonal elements (earlier copied into BD) are set to zero to avoid underflows which would otherwise occur.

The algorithm assumes after the initial origin shift that the matrix is positive definite, and so can expect diagonal elements of certain intermediate matrices to be positive. If any of these are negative, then IERR is flagged indicating that IDEF was set erroneously. Also, because the matrix is positive definite, the sum of the origin shifts must increase. If the sum does not increase, this is considered an irregular end of iteration and IERR is flagged. In this case, the current sum of the origin shifts is taken as an eigenvalue, and the matrix is deflated by deleting the row and column corresponding to the smallest diagonal element.

Finally, if TYPE is false, RATQR again reverses the signs of the elements of W.

This subroutine is a translation of the Algol procedure RATQR written and discussed in detail by Reinsch and Bauer (1).

4. REFERENCES.

1) Reinsch, C. and Bauer, F.L., Rational QR Transformation With Newton Shift for Symmetric Tridiagonal Matrices, Num. Math. 11,264-272 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/6, 257-265, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine RATQR is numerically stable when limited to the computation of a few eigenvalues that are not tightly clustered (1); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix.

```
      SUBROUTINE RATQR(N,EPS1,D,E,E2,M,W,IND,BD,TYPE,IDEF,IERR)
C
      INTEGER I,J,K,M,N,II,JJ,K1,IDEF,IERR,JDEF
      REAL D(N),E(N),E2(N),W(N),BD(N)
      REAL F,P,Q,R,S,EP,QP,ERR,TOT,EPS1,DELTA,MACHEP
      REAL ABS,AMIN1
      INTEGER IND(N)
      LOGICAL TYPE
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      JDEF = IDEF
C     ********** COPY D ARRAY INTO W **********
      DO 20 I = 1, N
   20 W(I) = D(I)
C
      IF (TYPE) GO TO 40
      J = 1
      GO TO 400
   40 ERR = 0.0
      S = 0.0
C     ********** LOOK FOR SMALL SUB-DIAGONAL ENTRIES AND DEFINE
C                INITIAL SHIFT FROM LOWER GERSCHGORIN BOUND.
C                COPY E2 ARRAY INTO BD **********
      TOT = W(1)
      Q = 0.0
      J = 0
C
      DO 100 I = 1, N
         P = Q
         IF (I .EQ. 1) GO TO 60
         IF (P .GT. MACHEP * (ABS(D(I)) + ABS(D(I-1)))) GO TO 80
   60    E2(I) = 0.0
   80    BD(I) = E2(I)
C     ********** COUNT ALSO IF ELEMENT OF E2 HAS UNDERFLOWED **********
         IF (E2(I) .EQ. 0.0) J = J + 1
         IND(I) = J
         Q = 0.0
         IF (I .NE. N) Q = ABS(E(I+1))
         TOT = AMIN1(W(I)-P-Q,TOT)
  100 CONTINUE
C
      IF (JDEF .EQ. 1 .AND. TOT .LT. 0.0) GO TO 140
C
      DO 110 I = 1, N
  110 W(I) = W(I) - TOT
C
      GO TO 160
  140 TOT = 0.0
```

```
C
  160 DO 360 K = 1, M
C     ********** NEXT QR TRANSFORMATION **********
  180    TOT = TOT + S
         DELTA = W(N) - S
         I = N
         F = ABS(MACHEP*TOT)
         IF (EPS1 .LT. F) EPS1 = F
         IF (DELTA .GT. EPS1) GO TO 190
         IF (DELTA .LT. (-EPS1)) GO TO 1000
         GO TO 300
C     ********** REPLACE SMALL SUB-DIAGONAL SQUARES BY ZERO
C                TO REDUCE THE INCIDENCE OF UNDERFLOWS **********
  190    IF (K .EQ. N) GO TO 210
         K1 = K + 1
         DO 200 J = K1, N
            IF (BD(J) .LE. (MACHEP*(W(J)+W(J-1))) ** 2) BD(J) = 0.0
  200    CONTINUE
C
  210    F = BD(N) / DELTA
         QP = DELTA + F
         P = 1.0
         IF (K .EQ. N) GO TO 260
         K1 = N - K
C     ********** FOR I=N-1 STEP -1 UNTIL K DO -- **********
         DO 240 II = 1, K1
            I = N - II
            Q = W(I) - S - F
            R = Q / QP
            P = P * R + 1.0
            EP = F * R
            W(I+1) = QP + EP
            DELTA = Q - EP
            IF (DELTA .GT. EPS1) GO TO 220
            IF (DELTA .LT. (-EPS1)) GO TO 1000
            GO TO 300
  220       F = BD(I) / Q
            QP = DELTA + F
            BD(I+1) = QP * EP
  240    CONTINUE
C
  260    W(K) = QP
         S = QP / P
         IF (TOT + S .GT. TOT) GO TO 180
C     ********** SET ERROR -- IRREGULAR END OF ITERATION.
C                DEFLATE MINIMUM DIAGONAL ELEMENT **********
         IERR = 5 * N + K
         S = 0.0
         DELTA = QP
```

```
C
         DO 280 J = K, N
            IF (W(J) .GT. DELTA) GO TO 280
            I = J
            DELTA = W(J)
  280    CONTINUE
C      ********** CONVERGENCE **********
  300    IF (I .LT. N) BD(I+1) = BD(I) * F / QP
         II = IND(I)
         IF (I .EQ. K) GO TO 340
         K1 = I - K
C      ********** FOR J=I-1 STEP -1 UNTIL K DO -- **********
         DO 320 JJ = 1, K1
            J = I - JJ
            W(J+1) = W(J) - S
            BD(J+1) = BD(J)
            IND(J+1) = IND(J)
  320    CONTINUE
C
  340    W(K) = TOT
         ERR = ERR + ABS(DELTA)
         BD(K) = ERR
         IND(K) = II
  360 CONTINUE
C
      IF (TYPE) GO TO 1001
      F = BD(1)
      E2(1) = 2.0
      BD(1) = F
      J = 2
C      ********** NEGATE ELEMENTS OF W FOR LARGEST VALUES **********
  400 DO 500 I = 1, N
  500 W(I) = -W(I)
C
      JDEF = -JDEF
      GO TO (40,1001), J
C      ********** SET ERROR -- IDEF SPECIFIED INCORRECTLY **********
 1000 IERR = 6 * N + 1
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F303 RG

A Fortran IV Driver Subroutine to Determine the Eigenvalues and Eigenvectors of a Real General Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine RG calls the recommended sequence of subroutines from the eigensystem subroutine package EISPACK to determine the eigenvalues and eigenvectors (if desired) of a real general matrix.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  RG(NM,N,A,WR,WI,MATZ,Z,IV1,FV1,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

N — is an integer input variable set equal to the order of the matrix A. N must not be greater than NM.

A — is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, A contains the real general matrix of order N whose eigenvalues and eigenvectors are to be found. On output, A has been destroyed.

WR,WI are working precision real output one-dimensional variables of dimension at least N containing the real and imaginary parts, respectively, of the eigenvalues of the real general matrix. The eigenvalues are unordered except that complex conjugate pairs of eigenvalues appear consecutively with the eigenvalue having the positive imaginary part first.

MATZ is an integer input variable set equal to zero if only eigenvalues are desired; otherwise it is set to any non-zero integer for both eigenvalues and eigenvectors.

Z is, if MATZ is non-zero, a working precision real output two-dimensional variable with row dimension NM and column dimension at least N containing the real and imaginary parts of the eigenvectors. If the J-th eigenvalue is real, the J-th column of Z contains its eigenvector. If the J-th eigenvalue is complex with positive imaginary part, the J-th and (J+1)-th columns of Z contain the real and imaginary parts of its eigenvector. The conjugate of this vector is the eigenvector for the conjugate eigenvalue. The eigenvectors are not normalized. If MATZ is zero, Z is not referenced and can be a dummy variable.

IV1 is an integer temporary one-dimensional variable of dimension at least N.

FV1 is a working precision temporary one-dimensional variable of dimension at least N.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If N is greater than NM, the subroutine terminates with IERR set equal to 10*N.

If more than 30 iterations are required to determine an eigenvalue, the subroutine terminates with IERR set equal to the index of the eigenvalue for which the failure occurs. The eigenvalues in the WR and WI arrays should be correct for indices IERR+1,IERR+2,...,N, but no eigenvectors are computed.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

This subroutine can be used to find all the eigenvalues and all the eigenvectors (if desired) of a real general matrix.

3. DISCUSSION OF METHOD AND ALGORITHM.

This subroutine calls the recommended sequence of subroutines from EISPACK to find the eigenvalues and eigenvectors of a real general matrix.

To find eigenvalues only, the sequence is the following.

- BALANC - to balance a real general matrix.
- ELMHES - to reduce the balanced matrix to an upper Hessenberg matrix using elementary transformations.
- HQR - to determine the eigenvalues of the original matrix from the Hessenberg matrix.

To find eigenvalues and eigenvectors, the sequence is the following.

- BALANC - to balance a real general matrix.
- ELMHES - to reduce the balanced matrix to an upper Hessenberg matrix using elementary transformations.
- ELTRAN - to accumulate the transformations from the Hessenberg reduction.
- HQR2 - to determine the eigenvalues and eigenvectors of the balanced matrix from the Hessenberg matrix.
- BALBAK - to backtransform the eigenvectors to those of the original matrix.

4. REFERENCES.

1) Garbow, B.S. and Dongarra, J.J., EISPACK Path Chart, April, 1974.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real general matrices.

B. Accuracy.

The accuracy of RG can be best described in terms of its role in those paths of EISPACK (1) which find eigenvalues and eigenvectors of real general matrices. In these paths, this subroutine is, in practice, numerically stable. This stability contributes to the property of these paths that each computed eigenvalue and its corresponding eigenvector are exact for a matrix close to the original matrix.

```
      SUBROUTINE RG(NM,N,A,WR,WI,MATZ,Z,IV1,FV1,IERR)
C
      INTEGER N,NM,IS1,IS2,IERR,MATZ
      REAL A(NM,N),WR(N),WI(N),Z(NM,N),FV1(N)
      INTEGER IV1(N)
C
      IF (N .LE. NM) GO TO 10
      IERR = 10 * N
      GO TO 50
C
   10 CALL  BALANC(NM,N,A,IS1,IS2,FV1)
      CALL  ELMHES(NM,N,IS1,IS2,A,IV1)
      IF (MATZ .NE. 0) GO TO 20
C     ********** FIND EIGENVALUES ONLY **********
      CALL  HQR(NM,N,IS1,IS2,A,WR,WI,IERR)
      GO TO 50
C     ********** FIND BOTH EIGENVALUES AND EIGENVECTORS **********
   20 CALL  ELTRAN(NM,N,IS1,IS2,A,IV1,Z)
      CALL  HQR2(NM,N,IS1,IS2,A,WR,WI,Z,IERR)
      IF (IERR .NE. 0) GO TO 50
      CALL  BALBAK(NM,N,IS1,IS2,FV1,N,Z)
   50 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F304 RS

A Fortran IV Driver Subroutine to Determine the Eigenvalues and Eigenvectors of a Real Symmetric Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine RS calls the recommended sequence of subroutines from the eigensystem subroutine package EISPACK to determine the eigenvalues and eigenvectors (if desired) of a real symmetric matrix.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  RS(NM,N,A,W,MATZ,Z,FV1,FV2,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

N — is an integer input variable set equal to the order of the matrix A. N must not be greater than NM.

A — is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, A contains the real symmetric matrix of order N whose eigenvalues and eigenvectors are to be found. Only the full lower triangle of A need be supplied. On output, the full upper triangle of A is unaltered.

W is a working precision real output one-dimensional variable of dimension at least N containing the eigenvalues of the real symmetric matrix in ascending order.

MATZ is an integer input variable set equal to zero if only eigenvalues are desired; otherwise it is set to any non-zero integer for both eigenvalues and eigenvectors.

Z is, if MATZ is non-zero, a working precision real output two-dimensional variable with row dimension NM and column dimension at least N containing the eigenvectors. The eigenvectors are orthonormal. If MATZ is zero, Z is not referenced and can be a dummy variable.

FV1,FV2 are working precision temporary one-dimensional variables of dimension at least N.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If N is greater than NM, the subroutine terminates with IERR set equal to 10*N.

If more than 30 iterations are required to determine an eigenvalue, the subroutine terminates with IERR set equal to the index of the eigenvalue for which the failure occurs. The eigenvalues and eigenvectors in the W and Z arrays should be correct for indices 1,2,...,IERR-1.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

This subroutine can be used to find all the eigenvalues and all the eigenvectors (if desired) of a real symmetric matrix.

3. DISCUSSION OF METHOD AND ALGORITHM.

This subroutine calls the recommended sequence of subroutines from EISPACK to find the eigenvalues and eigenvectors of a real symmetric matrix.

To find eigenvalues only, the sequence is the following.

- TRED1 - to reduce a real symmetric matrix to a symmetric tridiagonal matrix using orthogonal transformations.
- TQLRAT - to determine the eigenvalues of the original matrix from the symmetric tridiagonal matrix.

To find eigenvalues and eigenvectors, the sequence is the following.

- TRED2 - to reduce a real symmetric matrix to a symmetric tridiagonal matrix using and accumulating orthogonal transformations.
- TQL2 - to determine the eigenvalues and eigenvectors of the original matrix from the symmetric tridiagonal matrix.

4. REFERENCES.

1) Garbow, B.S. and Dongarra, J.J., EISPACK Path Chart, April, 1974.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real symmetric matrices.

B. Accuracy.

The accuracy of RS can best be described in terms of its role in those paths of EISPACK (1) which find eigenvalues and eigenvectors of real symmetric matrices. In these paths, this subroutine is numerically stable. This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE RS(NM,N,A,W,MATZ,Z,FV1,FV2,IERR)
C
      INTEGER N,NM,IERR,MATZ
      REAL A(NM,N),W(N),Z(NM,N),FV1(N),FV2(N)
C
      IF (N .LE. NM) GO TO 10
      IERR = 10 * N
      GO TO 50
C
   10 IF (MATZ .NE. 0) GO TO 20
C     ********** FIND EIGENVALUES ONLY **********
      CALL  TRED1(NM,N,A,W,FV1,FV2)
      CALL  TQLRAT(N,W,FV2,IERR)
      GO TO 50
C     ********** FIND BOTH EIGENVALUES AND EIGENVECTORS **********
   20 CALL  TRED2(NM,N,A,W,FV1,Z)
      CALL  TQL2(NM,N,W,FV1,Z,IERR)
   50 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F307 RSP

A Fortran IV Driver Subroutine to Determine the Eigenvalues and Eigenvectors of a Real Symmetric Packed Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine RSP calls the recommended sequence of subroutines from the eigensystem subroutine package EISPACK to determine the eigenvalues and eigenvectors (if desired) of a real symmetric packed matrix.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  RSP(NM,N,NV,A,W,MATZ,Z,FV1,FV2,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array Z as specified in the DIMENSION statement for Z in the calling program.

N — is an integer input variable set equal to the order of the matrix A. N must not be greater than NM.

NV — is an integer input variable set equal to the dimension of the array A as specified in the DIMENSION statement for A in the calling program. NV must not be less than N*(N+1)/2.

A is a working precision real one-dimensional variable of dimension at least NV. On input, A contains the lower triangle, stored row-wise, of the real symmetric packed matrix of order N whose eigenvalues and eigenvectors are to be found. For example if N=3, A should contain

(A(1,1),A(2,1),A(2,2),A(3,1),A(3,2),A(3,3))

where the subscripts for each element refer to the row and column of the element in the standard two-dimensional representation. On output, A has been destroyed.

W is a working precision real output one-dimensional variable of dimension at least N containing the eigenvalues of the real symmetric packed matrix in ascending order.

MATZ is an integer input variable set equal to zero if only eigenvalues are desired; otherwise it is set to any non-zero integer for both eigenvalues and eigenvectors.

Z is, if MATZ is non-zero, a working precision real output two-dimensional variable with row dimension NM and column dimension at least N containing the eigenvectors. The eigenvectors are orthonormal. If MATZ is zero, Z is not referenced and can be a dummy variable.

FV1,FV2 are working precision temporary one-dimensional variables of dimension at least N.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If N is greater than NM, the subroutine terminates with IERR set equal to 10*N.

If NV is less than N*(N+1)/2, the subroutine terminates with IERR set equal to 20*N.

If more than 30 iterations are required to determine an eigenvalue, the subroutine terminates with IERR set equal to the index of the eigenvalue for which the failure occurs. The eigenvalues and eigenvectors in the W and Z arrays should be correct for indices 1,2,...,IERR-1.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

This subroutine can be used to find all the eigenvalues and all the eigenvectors (if desired) of a real symmetric packed matrix.

3. DISCUSSION OF METHOD AND ALGORITHM.

This subroutine calls the recommended sequence of subroutines from EISPACK to find the eigenvalues and eigenvectors of a real symmetric packed matrix.

To find eigenvalues only, the sequence is the following.

- TRED3 - to reduce a real symmetric packed matrix to a symmetric tridiagonal matrix using orthogonal transformations.
- TQLRAT - to determine the eigenvalues of the original matrix from the symmetric tridiagonal matrix.

To find eigenvalues and eigenvectors, the sequence is the following.

- TRED3 - to reduce a real symmetric packed matrix to a symmetric tridiagonal matrix using orthogonal transformations.
- TQL2 - to determine the eigenvalues and eigenvectors of the symmetric tridiagonal matrix.
- TRBAK3 - to backtransform the eigenvectors to those of the original matrix.

4. REFERENCES.

1) Garbow, B.S. and Dongarra, J.J., EISPACK Path Chart, April, 1974.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real symmetric packed matrices.

B. Accuracy.

The accuracy of RSP can best be described in terms of its role in those paths of EISPACK (1) which find eigenvalues and eigenvectors of real symmetric packed matrices. In these paths, this subroutine is numerically stable. This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE RSP(NM,N,NV,A,W,MATZ,Z,FV1,FV2,IERR)
C
      INTEGER I,J,N,NM,NV,IERR,MATZ
      REAL A(NV),W(N),Z(NM,N),FV1(N),FV2(N)
C
      IF (N .LE. NM) GO TO 5
      IERR = 10 * N
      GO TO 50
    5 IF (NV .GE. (N * (N + 1)) / 2) GO TO 10
      IERR = 20 * N
      GO TO 50
C
   10 CALL  TRED3(N,NV,A,W,FV1,FV2)
      IF (MATZ .NE. 0) GO TO 20
C     ********** FIND EIGENVALUES ONLY **********
      CALL  TQLRAT(N,W,FV2,IERR)
      GO TO 50
C     ********** FIND BOTH EIGENVALUES AND EIGENVECTORS **********
   20 DO 40 I = 1, N
C
         DO 30 J = 1, N
            Z(J,I) = 0.0
   30    CONTINUE
C
         Z(I,I) = 1.0
   40 CONTINUE
C
      CALL  TQL2(NM,N,W,FV1,Z,IERR)
      IF (IERR .NE. 0) GO TO 50
      CALL  TRBAK3(NM,N,NV,A,N,Z)
   50 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F305 RST

A Fortran IV Driver Subroutine to Determine the Eigenvalues and Eigenvectors of a Real Symmetric Tridiagonal Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine RST calls the recommended sequence of subroutines from the eigensystem subroutine package EISPACK to determine the eigenvalues and eigenvectors (if desired) of a real symmetric tridiagonal matrix.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  RST(NM,N,W,E,MATZ,Z,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array Z as specified in the DIMENSION statement for Z in the calling program.

N — is an integer input variable set equal to the order of the matrix. N must not be greater than NM.

W — is a working precision real one-dimensional variable of dimension at least N. On input, it contains the diagonal elements of the real symmetric tridiagonal matrix. On output, it contains the eigenvalues of the matrix in ascending order.

E — is a working precision real one-dimensional variable of dimension at least N. On input, it contains the subdiagonal elements of the matrix in its last N-1 positions. E(1) is arbitrary. On output, E has been destroyed.

MATZ — is an integer input variable set equal to zero if only eigenvalues are desired; otherwise it is set to any non-zero integer for both eigenvalues and eigenvectors.

Z — is, if MATZ is non-zero, a working precision real output two-dimensional variable with row dimension NM and column dimension at least N containing the eigenvectors. The eigenvectors are orthonormal. If MATZ is zero, Z is not referenced and can be a dummy variable.

IERR — is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If N is greater than NM, the subroutine terminates with IERR set equal to 10*N.

If more than 30 iterations are required to determine an eigenvalue, the subroutine terminates with IERR set equal to the index of the eigenvalue for which the failure occurs. The eigenvalues and eigenvectors in the W and Z arrays should be correct for indices 1,2,...,IERR-1.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

This subroutine can be used to find all the eigenvalues and all the eigenvectors (if desired) of a real symmetric tridiagonal matrix.

3. DISCUSSION OF METHOD AND ALGORITHM.

This subroutine calls the recommended sequence of subroutines from EISPACK to find the eigenvalues and eigenvectors of a real symmetric tridiagonal matrix.

To find eigenvalues only, the sequence is the following.

IMTQL1 - to determine the eigenvalues of a real symmetric tridiagonal matrix.

To find eigenvalues and eigenvectors, the sequence is the following.

IMTQL2 - to determine the eigenvalues and eigenvectors of a real symmetric tridiagonal matrix.

4. REFERENCES.

1) Garbow, B.S. and Dongarra, J.J., EISPACK Path Chart, April, 1974.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real symmetric tridiagonal matrices.

B. Accuracy.

The subroutine RST is numerically stable; that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE RST(NM,N,W,E,MATZ,Z,IERR)
C
      INTEGER I,J,N,NM,IERR,MATZ
      REAL W(N),E(N),Z(NM,N)
C
      IF (N .LE. NM) GO TO 10
      IERR = 10 * N
      GO TO 50
C
   10 IF (MATZ .NE. 0) GO TO 20
C     ********** FIND EIGENVALUES ONLY **********
      CALL  IMTQL1(N,W,E,IERR)
      GO TO 50
C     ********** FIND BOTH EIGENVALUES AND EIGENVECTORS **********
   20 DO 40 I = 1, N
C
         DO 30 J = 1, N
            Z(J,I) = 0.0
   30    CONTINUE
C
         Z(I,I) = 1.0
   40 CONTINUE
C
      CALL  IMTQL2(NM,N,W,E,Z,IERR)
   50 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F306 RT

A Fortran IV Driver Subroutine to Determine the Eigenvalues and Eigenvectors of a Certain Real Tridiagonal Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine RT calls the recommended sequence of subroutines from the eigensystem subroutine package EISPACK to determine the eigenvalues and eigenvectors (if desired) of a certain real tridiagonal matrix. The property of the matrix required for use of this subroutine is that the products of pairs of corresponding off-diagonal elements be all non-negative, and further if eigenvectors are desired, no product be zero unless both factors are zero.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  RT(NM,N,A,W,MATZ,Z,FV1,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

N — is an integer input variable set equal to the order of the matrix A. N must not be greater than NM.

A is a working precision real input two-dimensional variable with row dimension NM and column dimension at least 3 containing the real tridiagonal matrix of order N whose eigenvalues and eigenvectors are to be found. Its subdiagonal elements are stored in the last N-1 positions of the first column, its diagonal elements are stored in the second column, and its superdiagonal elements are stored in the first N-1 positions of the third column. Elements A(1,1) and A(N,3) are arbitrary.

W is a working precision real output one-dimensional variable of dimension at least N containing the eigenvalues of the real tridiagonal matrix in ascending order.

MATZ is an integer input variable set equal to zero if only eigenvalues are desired; otherwise it is set to any non-zero integer for both eigenvalues and eigenvectors.

Z is, if MATZ is non-zero, a working precision real output two-dimensional variable with row dimension NM and column dimension at least N containing the eigenvectors. The eigenvectors are not normalized. If MATZ is zero, Z is not referenced and can be a dummy variable.

FV1 is a working precision temporary one-dimensional variable of dimension at least N.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If N is greater than NM, the subroutine terminates with IERR set equal to 10*N.

If the product of A(I,1) and A(I-1,3) is negative, the subroutine terminates with IERR set equal to N+I and no results computed.

If the product is zero with one factor non-zero, and MATZ is non-zero, the subroutine terminates with IERR set equal to 2*N+I and no results computed.

If more than 30 iterations are required to determine an eigenvalue, the subroutine terminates with IERR set equal to the index of the eigenvalue for which the failure occurs. The eigenvalues and eigenvectors in the W and Z arrays should be correct for indices 1,2,...,IERR-1.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

This subroutine can be used to find all the eigenvalues and all the eigenvectors (if desired) of a certain real tridiagonal matrix.

3. DISCUSSION OF METHOD AND ALGORITHM.

This subroutine calls the recommended sequence of subroutines from EISPACK to find the eigenvalues and eigenvectors of a certain real tridiagonal matrix.

To find eigenvalues only, the sequence is the following.

- FIGI - to transform a certain real tridiagonal matrix into a symmetric tridiagonal matrix.
- IMTQL1 - to determine the eigenvalues of the original matrix from the symmetric tridiagonal matrix.

To find eigenvalues and eigenvectors, the sequence is the following.

- FIGI2 - to transform a certain real tridiagonal matrix into a symmetric tridiagonal matrix, accumulating the transformations.
- IMTQL2 - to determine the eigenvalues and eigenvectors of the original matrix from the symmetric tridiagonal matrix.

4. REFERENCES.

1) Garbow, B.S. and Dongarra, J.J., EISPACK Path Chart, April, 1974.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The accuracy of RT can best be described in terms of its role in those paths of EISPACK (1) which find eigenvalues and eigenvectors of certain real non-symmetric tridiagonal matrices. In these paths, this subroutine is numerically stable. This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE RT(NM,N,A,W,MATZ,Z,FV1,IERR)
C
      INTEGER N,NM,IERR,MATZ
      REAL A(NM,3),W(N),Z(NM,N),FV1(N)
C
      IF (N .LE. NM) GO TO 10
      IERR = 10 * N
      GO TO 50
C
   10 IF (MATZ .NE. 0) GO TO 20
C     ********** FIND EIGENVALUES ONLY **********
      CALL  FIGI(NM,N,A,W,FV1,FV1,IERR)
      IF (IERR .GT. 0) GO TO 50
      CALL  IMTQL1(N,W,FV1,IERR)
      GO TO 50
C     ********** FIND BOTH EIGENVALUES AND EIGENVECTORS **********
   20 CALL  FIGI2(NM,N,A,W,FV1,Z,IERR)
      IF (IERR .NE. 0) GO TO 50
      CALL  IMTQL2(NM,N,W,FV1,Z,IERR)
   50 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F223-2 TINVIT

A Fortran IV Subroutine to Determine Some Eigenvectors of a Symmetric Tridiagonal Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine TINVIT determines those eigenvectors of a symmetric tridiagonal matrix corresponding to a set of ordered approximate eigenvalues, using inverse iteration.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  TINVIT(NM,N,D,E,E2,M,W,IND,Z,
                   IERR,RV1,RV2,RV3,RV4,RV6)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array Z as specified in the DIMENSION statement for Z in the calling program.

N — is an integer input variable set equal to the order of the matrix. N must be not greater than NM.

D — is a working precision real input one-dimensional variable of dimension at least N containing the diagonal elements of the symmetric tridiagonal matrix.

E is a working precision real input one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary.

E2 is a working precision real input one-dimensional variable of dimension at least N containing, in its last N-1 positions, the squares of the corresponding elements of E with zeros corresponding to negligible elements of E. (The successive submatrices are located from the zeros of E2.) If MACHEP denotes the relative machine precision, then E(I) is considered negligible if it is not larger than the product of MACHEP and the sum of the magnitudes of D(I) and D(I-1). E2(1) should contain 0.0 if the eigenvalues are in ascending order and 2.0 if the eigenvalues are in descending order. If subroutine BISECT (F294) or TRIDIB (F237) has been used to determine the eigenvalues, their output E2 array is suitable for input to TINVIT.

M is an integer input variable set equal to the number of specified eigenvalues for which the corresponding eigenvectors are to be determined.

W is a working precision real input one-dimensional variable of dimension at least M containing the M specified eigenvalues of the symmetric tridiagonal matrix. The eigenvalues must be in either ascending or descending order in W. The ordering is required to insure the determination of independent orthogonal eigenvectors associated with close eigenvalues.

IND is an integer input one-dimensional variable of dimension at least M containing the submatrix indices associated with the corresponding M eigenvalues in W. Eigenvalues belonging to the first submatrix have index 1, those belonging to the second submatrix have index 2, etc. If BISECT or TRIDIB has been used to determine the eigenvalues, their output IND array is suitable for input to TINVIT.

Z is a working precision real output two-dimensional variable with row dimension NM and column dimension at least M. It contains M orthonormal eigenvectors of the symmetric tridiagonal matrix corresponding to the M eigenvalues in W.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

RV1,RV2,RV3
are working precision real temporary one-dimensional variables of dimension at least N used to store the main diagonal and the two adjacent diagonals of the triangular matrix produced in the inverse iteration process.

RV4,RV6 are working precision real temporary one-dimensional variables of dimension at least N. RV4 holds the multipliers of the Gaussian elimination step in the inverse iteration process. RV6 holds the approximate eigenvectors in this process.

B. Error Conditions and Returns.

If more than 5 iterations are required to determine an eigenvector, TINVIT terminates the computation for that eigenvector and sets IERR to -R where R is the index of the eigenvector. If this failure occurs for more than one eigenvector, the last occurrence is recorded in IERR. The columns of Z corresponding to failures of the above sort are set to zero vectors.

If all the eigenvectors are determined within 5 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine some of the eigenvalues and eigenvectors of a full symmetric matrix, TINVIT should be preceded by TRED1 (F277) to provide a suitable symmetric tridiagonal matrix for BISECT (F294), TRIDIB (F237), or IMTQLV (F234) which can then be used to determine the eigenvalues. It should be followed by TRBAK1 (F279) to back transform the eigenvectors from TINVIT into those of the original matrix.

To determine some of the eigenvalues and eigenvectors of a complex Hermitian matrix, TINVIT should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for BISECT (F294), TRIDIB (F237), or IMTQLV (F234). It should then be followed by HTRIBK (F285) to back transform the eigenvectors from TINVIT into those of the original matrix.

Some of the eigenvalues and eigenvectors of certain non-symmetric tridiagonal matrices can be computed using the combination of FIGI (F280), BISECT (F294) or TRIDIB (F237), TINVIT, and BAKVEC (F281). See F280 for the description of this special class of matrices. For these matrices, TINVIT should be preceded by FIGI to provide a suitable symmetric matrix for BISECT or TRIDIB. It should then be followed by BAKVEC to back transform the eigenvectors from TINVIT into those of the original matrix.

The computation of the eigenvectors by inverse iteration requires that the precision of the eigenvalues be commensurate with small relative perturbations of the order of MACHEP in the matrix elements. For most symmetric tridiagonal matrices, it is enough that the absolute error in the eigenvalues for which eigenvectors are desired be approximately MACHEP times a norm of the matrix. But some matrices require a smaller absolute error, perhaps as small as MACHEP times the eigenvalue of smallest magnitude.

3. DISCUSSION OF METHOD AND ALGORITHM.

The calculations proceed as follows. First, the E2 array is inspected for the presence of a zero element defining a submatrix. The eigenvalues belonging to this submatrix are identified by their common submatrix index in IND.

The eigenvectors of the submatrix are then computed by inverse iteration. First, the LU decomposition of the submatrix with an approximate eigenvalue subtracted from its diagonal elements is achieved by Gaussian elimination using partial pivoting. The multipliers defining the lower triangular matrix L are stored in the temporary array RV4 and the upper triangular matrix U is stored in the three temporary arrays RV1, RV2, and RV3. Saving these quantities in RV1, RV2, RV3, and RV4 avoids repeating the LU decomposition if further iterations are required. An approximate vector, stored in RV6, is computed starting from an initial vector, and the norm of the approximate vector is compared with a norm of the submatrix to determine whether the growth is sufficient to accept it as an eigenvector. If this vector is accepted, its Euclidean norm

is made 1. If the growth is not sufficient, this vector is used as the initial vector in computing the next approximate vector. This iteration process is repeated at most 5 times.

Eigenvectors computed in the above way corresponding to well-separated eigenvalues of this submatrix will be orthogonal. However, eigenvectors corresponding to close eigenvalues of this submatrix may not be satisfactorily orthogonal. Hence, to insure orthogonal eigenvectors, each approximate vector is made orthogonal to those previously computed eigenvectors whose eigenvalues are close to the current eigenvalue. If the orthogonalization process produces a zero vector, a column of the identity matrix is used as an initial vector for the next iteration.

Identical eigenvalues are perturbed slightly in an attempt to obtain independent eigenvectors. These perturbations are not recorded in the eigenvalue array W.

The above steps are repeated on each submatrix until all the eigenvectors are computed.

This subroutine is a subset (except for the resolution of non-convergent eigenvectors) of the Fortran subroutine TSTURM (F293), which is a translation of the Algol procedure TRISTURM written and discussed in detail by Peters and Wilkinson (1).

4. REFERENCES.

1) Peters, G. and Wilkinson, J.H., The Calculation of Specified Eigenvectors by Inverse Iteration, Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/18, 418-439, Springer-Verlag, 1971.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The accuracy of TINVIT can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real symmetric matrices and matrix systems. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix or system close to the original matrix or system and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix or system.

```
      SUBROUTINE TINVIT(NM,N,D,E,E2,M,W,IND,Z,
     X                  IERR,RV1,RV2,RV3,RV4,RV6)
C
      INTEGER I,J,M,N,P,Q,R,S,II,IP,JJ,NM,ITS,TAG,IERR,GROUP
      REAL D(N),E(N),E2(N),W(M),Z(NM,M),
     X       RV1(N),RV2(N),RV3(N),RV4(N),RV6(N)
      REAL U,V,UK,XU,X0,X1,EPS2,EPS3,EPS4,NORM,ORDER,MACHEP
      REAL SQRT,ABS,FLOAT
      INTEGER IND(M)
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      IF (M .EQ. 0) GO TO 1001
      TAG = 0
      ORDER = 1.0 - E2(1)
      Q = 0
C     ********** ESTABLISH AND PROCESS NEXT SUBMATRIX **********
  100 P = Q + 1
C
      DO 120 Q = P, N
         IF (Q .EQ. N) GO TO 140
         IF (E2(Q+1) .EQ. 0.0) GO TO 140
  120 CONTINUE
C     ********** FIND VECTORS BY INVERSE ITERATION **********
  140 TAG = TAG + 1
      S = 0
C
      DO 920 R = 1, M
         IF (IND(R) .NE. TAG) GO TO 920
         ITS = 1
         X1 = W(R)
         IF (S .NE. 0) GO TO 510
C     ********** CHECK FOR ISOLATED ROOT **********
         XU = 1.0
         IF (P .NE. Q) GO TO 490
         RV6(P) = 1.0
         GO TO 870
  490    NORM = ABS(D(P))
         IP = P + 1
C
         DO 500 I = IP, Q
  500    NORM = NORM + ABS(D(I)) + ABS(E(I))
C     ********** EPS2 IS THE CRITERION FOR GROUPING,
C                EPS3 REPLACES ZERO PIVOTS AND EQUAL
C                ROOTS ARE MODIFIED BY EPS3,
C                EPS4 IS TAKEN VERY SMALL TO AVOID OVERFLOW **********
         EPS2 = 1.0E-3 * NORM
         EPS3 = MACHEP * NORM
         UK = FLOAT(Q-P+1)
```

```
         EPS4 = UK * EPS3
         UK = EPS4 / SQRT(UK)
         S = P
  505    GROUP = 0
         GO TO 520
C     ********** LOOK FOR CLOSE OR COINCIDENT ROOTS **********
  510    IF (ABS(X1-X0) .GE. EPS2) GO TO 505
         GROUP = GROUP + 1
         IF (ORDER * (X1 - X0) .LE. 0.0) X1 = X0 + ORDER * EPS3
C     ********** ELIMINATION WITH INTERCHANGES AND
C                INITIALIZATION OF VECTOR **********
  520    V = 0.0
C
         DO 580 I = P, Q
            RV6(I) = UK
            IF (I .EQ. P) GO TO 560
            IF (ABS(E(I)) .LT. ABS(U)) GO TO 540
C     ********** WARNING -- A DIVIDE CHECK MAY OCCUR HERE IF
C                E2 ARRAY HAS NOT BEEN SPECIFIED CORRECTLY **********
            XU = U / E(I)
            RV4(I) = XU
            RV1(I-1) = E(I)
            RV2(I-1) = D(I) - X1
            RV3(I-1) = 0.0
            IF (I .NE. Q) RV3(I-1) = E(I+1)
            U = V - XU * RV2(I-1)
            V = -XU * RV3(I-1)
            GO TO 580
  540       XU = E(I) / U
            RV4(I) = XU
            RV1(I-1) = U
            RV2(I-1) = V
            RV3(I-1) = 0.0
  560       U = D(I) - X1 - XU * V
            IF (I .NE. Q) V = E(I+1)
  580    CONTINUE
C
         IF (U .EQ. 0.0) U = EPS3
         RV1(Q) = U
         RV2(Q) = 0.0
         RV3(Q) = 0.0
C     ********** BACK SUBSTITUTION
C                FOR I=Q STEP -1 UNTIL P DO -- **********
  600    DO 620 II = P, Q
            I = P + Q - II
            RV6(I) = (RV6(I) - U * RV2(I) - V * RV3(I)) / RV1(I)
            V = U
            U = RV6(I)
  620    CONTINUE
C     ********** ORTHOGONALIZE WITH RESPECT TO PREVIOUS
C                MEMBERS OF GROUP **********
         IF (GROUP .EQ. 0) GO TO 700
         J = R
```

```
C
      DO 680 JJ = 1, GROUP
  630    J = J - 1
         IF (IND(J) .NE. TAG) GO TO 630
         XU = 0.0
C
         DO 640 I = P, Q
  640    XU = XU + RV6(I) * Z(I,J)
C
         DO 660 I = P, Q
  660    RV6(I) = RV6(I) - XU * Z(I,J)
C
  680 CONTINUE
C
  700 NORM = 0.0
C
      DO 720 I = P, Q
  720 NORM = NORM + ABS(RV6(I))
C
      IF (NORM .GE. 1.0) GO TO 840
C     ********** FORWARD SUBSTITUTION **********
      IF (ITS .EQ. 5) GO TO 830
      IF (NORM .NE. 0.0) GO TO 740
      RV6(S) = EPS4
      S = S + 1
      IF (S .GT. Q) S = P
      GO TO 780
  740 XU = EPS4 / NORM
C
      DO 760 I = P, Q
  760 RV6(I) = RV6(I) * XU
C     ********** ELIMINATION OPERATIONS ON NEXT VECTOR
C                ITERATE **********
  780 DO 820 I = IP, Q
         U = RV6(I)
C     ********** IF RV1(I-1) .EQ. E(I), A ROW INTERCHANGE
C                WAS PERFORMED EARLIER IN THE
C                TRIANGULARIZATION PROCESS **********
         IF (RV1(I-1) .NE. E(I)) GO TO 800
         U = RV6(I-1)
         RV6(I-1) = RV6(I)
  800    RV6(I) = U - RV4(I) * RV6(I-1)
  820 CONTINUE
C
      ITS = ITS + 1
      GO TO 600
C     ********** SET ERROR -- NON-CONVERGED EIGENVECTOR **********
  830 IERR = -R
      XU = 0.0
      GO TO 870
C     ********** NORMALIZE SO THAT SUM OF SQUARES IS
C                1 AND EXPAND TO FULL ORDER **********
  840 U = 0.0
```

```
C
          DO 860 I = P, Q
  860     U = U + RV6(I)**2
C
          XU = 1.0 / SQRT(U)
C
  870     DO 880 I = 1, N
  880     Z(I,R) = 0.0
C
          DO 900 I = P, Q
  900     Z(I,R) = RV6(I) * XU
C
          X0 = X1
  920 CONTINUE
C
      IF (Q .LT. N) GO TO 100
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F235 TQLRAT

A Fortran IV Subroutine to Determine the Eigenvalues of a Symmetric Tridiagonal Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine TQLRAT determines the eigenvalues of a symmetric tridiagonal matrix using a rational variant of the QL method.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE TQLRAT(N,D,E2,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

N — is an integer input variable set equal to the order of the matrix.

D — is a working precision real one-dimensional variable of dimension at least N. On input, it contains the diagonal elements of the symmetric tridiagonal matrix. On output, it contains the eigenvalues of this matrix in ascending order.

E2 — is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the squares of the subdiagonal elements of the symmetric tridiagonal matrix. E2(1) is arbitrary. Note that TQLRAT destroys E2.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, TQLRAT terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the D array should be correct for indices 1,2,...,IERR-1. These eigenvalues are ordered but are not necessarily the smallest IERR-1 eigenvalues.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine the eigenvalues of a full symmetric matrix, TQLRAT should be preceded by TRED1 (F277) to provide a suitable symmetric tridiagonal matrix for TQLRAT.

To determine the eigenvalues of a complex Hermitian matrix, TQLRAT should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for TQLRAT.

TQLRAT does not perform well on matrices whose successive row sums vary widely in magnitude and are not strictly increasing from the first to the last row. The subroutine IMTQL1 (F291) is not sensitive to such row sums and is therefore recommended for symmetric tridiagonal matrices whose structure is not known.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by a rational variant of the QL method. The essence of this method is a process whereby a sequence of symmetric tridiagonal matrices, unitarily similar to the original symmetric tridiagonal matrix, is formed which converges to a diagonal matrix. The rate of convergence of this sequence is improved by shifting the origin at each iteration. Before the iterations for each eigenvalue, the symmetric tridiagonal matrix is checked for a possible splitting into submatrices. If a splitting occurs, only the uppermost submatrix participates in the next iteration. The eigenvalues are ordered in ascending order as they are found.

The origin shift at each iteration is the eigenvalue of the current uppermost 2x2 principal minor closer to the first diagonal element of this minor. Whenever the uppermost 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision.

This subroutine is a translation of the Algol procedure TQLRAT written and discussed in detail by Reinsch (1). It is a rational variant of the subroutine TQL1 which is a translation of the Algol procedure TQL1 written and discussed in detail by Bowdler, Martin, Reinsch, and Wilkinson (2).

4. REFERENCES.

1) Reinsch, C.H., A Stable Rational QR Algorithm for the Computation of the Eigenvalues of an Hermitian, Tridiagonal Matrix, Math. Comp. 25,591-597 (1971). (Algorithm 464, Comm. ACM 16,689 (1973).)

2) Bowdler, H., Martin, R.S., Reinsch, C., and Wilkinson, J.H., The QR and QL Algorithms for Symmetric Matrices, Num. Math. 11,293-306 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/3, 227-240, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine TQLRAT is numerically stable (1,2); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix.

```
      SUBROUTINE TQLRAT(N,D,E2,IERR)
C
      INTEGER I,J,L,M,N,II,L1,MML,IERR
      REAL D(N),E2(N)
      REAL B,C,F,G,H,P,R,S,MACHEP
      REAL SQRT,ABS,SIGN
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      IF (N .EQ. 1) GO TO 1001
C
      DO 100 I = 2, N
  100 E2(I-1) = E2(I)
C
      F = 0.0
      B = 0.0
      E2(N) = 0.0
C
      DO 290 L = 1, N
         J = 0
         H = MACHEP * (ABS(D(L)) + SQRT(E2(L)))
         IF (B .GT. H) GO TO 105
         B = H
         C = B * B
C     ********** LOOK FOR SMALL SQUARED SUB-DIAGONAL ELEMENT **********
  105    DO 110 M = L, N
            IF (E2(M) .LE. C) GO TO 120
C     ********** E2(N) IS ALWAYS ZERO, SO THERE IS NO EXIT
C                THROUGH THE BOTTOM OF THE LOOP **********
  110    CONTINUE
C
  120    IF (M .EQ. L) GO TO 210
  130    IF (J .EQ. 30) GO TO 1000
         J = J + 1
C     ********** FORM SHIFT **********
         L1 = L + 1
         S = SQRT(E2(L))
         G = D(L)
         P = (D(L1) - G) / (2.0 * S)
         R = SQRT(P*P+1.0)
         D(L) = S / (P + SIGN(R,P))
         H = G - D(L)
C
         DO 140 I = L1, N
  140    D(I) = D(I) - H
C
         F = F + H
```

```
C      ********** RATIONAL QL TRANSFORMATION **********
          G = D(M)
          IF (G .EQ. 0.0) G = B
          H = G
          S = 0.0
          MML = M - L
C      ********** FOR I=M-1 STEP -1 UNTIL L DO -- **********
          DO 200 II = 1, MML
             I = M - II
             P = G * H
             R = P + E2(I)
             E2(I+1) = S * R
             S = E2(I) / R
             D(I+1) = H + S * (H + D(I))
             G = D(I) - E2(I) / G
             IF (G .EQ. 0.0) G = B
             H = G * P / R
  200     CONTINUE
C
          E2(L) = S * G
          D(L) = H
C      ********** GUARD AGAINST UNDERFLOW IN CONVERGENCE TEST **********
          IF (H .EQ. 0.0) GO TO 210
          IF (ABS(E2(L)) .LE. ABS(C/H)) GO TO 210
          E2(L) = H * E2(L)
          IF (E2(L) .NE. 0.0) GO TO 130
  210     P = D(L) + F
C      ********** ORDER EIGENVALUES **********
          IF (L .EQ. 1) GO TO 250
C      ********** FOR I=L STEP -1 UNTIL 2 DO -- **********
          DO 230 II = 2, L
             I = L + 2 - II
             IF (P .GE. D(I-1)) GO TO 270
             D(I) = D(I-1)
  230     CONTINUE
C
  250     I = 1
  270     D(I) = P
  290 CONTINUE
C
      GO TO 1001
C      ********** SET ERROR -- NO CONVERGENCE TO AN
C                 EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = L
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F289-2 TQL1

A Fortran IV Subroutine to Determine the Eigenvalues of a Symmetric Tridiagonal Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine TQL1 determines the eigenvalues of a symmetric tridiagonal matrix using the QL method.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE TQL1(N,D,E,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

N — is an integer input variable set equal to the order of the matrix.

D — is a working precision real one-dimensional variable of dimension at least N. On input, it contains the diagonal elements of the symmetric tridiagonal matrix. On output, it contains the eigenvalues of this matrix in ascending order.

E — is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary. Note that TQL1 destroys E.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, TQL1 terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues in the D array should be correct for indices 1,2,...,IERR-1. These eigenvalues are ordered but are not necessarily the smallest IERR-1 eigenvalues.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine the eigenvalues of a full symmetric matrix, TQL1 should be preceded by TRED1 (F277) to provide a suitable symmetric tridiagonal matrix for TQL1.

To determine the eigenvalues of a complex Hermitian matrix, TQL1 should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for TQL1.

TQL1 does not perform well on matrices whose successive row sums vary widely in magnitude and are not strictly increasing from the first to the last row. The subroutine IMTQL1 (F291) is not sensitive to such row sums and is therefore recommended for symmetric tridiagonal matrices whose structure is not known.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the QL method. The essence of this method is a process whereby a sequence of symmetric tridiagonal matrices, unitarily similar to the original symmetric tridiagonal matrix, is formed which converges to a diagonal matrix. The rate of convergence of this sequence is improved by shifting the origin at each iteration. Before the iterations for each eigenvalue, the symmetric tridiagonal matrix is checked for a possible splitting into submatrices. If a splitting occurs, only the uppermost submatrix participates in the next iteration. The eigenvalues are ordered in ascending order as they are found.

The origin shift at each iteration is the eigenvalue of the current uppermost 2x2 principal minor closer to the first diagonal element of this minor. Whenever the uppermost 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision.

This subroutine is a translation of the Algol procedure TQL1 written and discussed in detail by Bowdler, Martin, Reinsch, and Wilkinson (1).

4. REFERENCES.

1) Bowdler, H., Martin, R.S., Reinsch, C., and Wilkinson, J.H., The QR and QL Algorithms for Symmetric Matrices, Num. Math. 11,293-306 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/3, 227-240, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine TQL1 is numerically stable (1); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix.

```
      SUBROUTINE TQL1(N,D,E,IERR)
C
      INTEGER I,J,L,M,N,II,L1,MML,IERR
      REAL D(N),E(N)
      REAL B,C,F,G,H,P,R,S,MACHEP
      REAL SQRT,ABS,SIGN
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      IF (N .EQ. 1) GO TO 1001
C
      DO 100 I = 2, N
  100 E(I-1) = E(I)
C
      F = 0.0
      B = 0.0
      E(N) = 0.0
C
      DO 290 L = 1, N
         J = 0
         H = MACHEP * (ABS(D(L)) + ABS(E(L)))
         IF (B .LT. H) B = H
C     ********** LOOK FOR SMALL SUB-DIAGONAL ELEMENT **********
         DO 110 M = L, N
            IF (ABS(E(M)) .LE. B) GO TO 120
C     ********** E(N) IS ALWAYS ZERO, SO THERE IS NO EXIT
C                THROUGH THE BOTTOM OF THE LOOP **********
  110    CONTINUE
C
  120    IF (M .EQ. L) GO TO 210
  130    IF (J .EQ. 30) GO TO 1000
         J = J + 1
C     ********** FORM SHIFT **********
         L1 = L + 1
         G = D(L)
         P = (D(L1) - G) / (2.0 * E(L))
         R = SQRT(P*P+1.0)
         D(L) = E(L) / (P + SIGN(R,P))
         H = G - D(L)
C
         DO 140 I = L1, N
  140    D(I) = D(I) - H
C
         F = F + H
C     ********** QL TRANSFORMATION **********
         P = D(M)
         C = 1.0
         S = 0.0
         MML = M - L
```

```
C     ********** FOR I=M-1 STEP -1 UNTIL L DO -- **********
         DO 200 II = 1, MML
            I = M - II
            G = C * E(I)
            H = C * P
            IF (ABS(P) .LT. ABS(E(I))) GO TO 150
            C = E(I) / P
            R = SQRT(C*C+1.0)
            E(I+1) = S * P * R
            S = C / R
            C = 1.0 / R
            GO TO 160
  150       C = P / E(I)
            R = SQRT(C*C+1.0)
            E(I+1) = S * E(I) * R
            S = 1.0 / R
            C = C * S
  160       P = C * D(I) - S * G
            D(I+1) = H + S * (C * G + S * D(I))
  200    CONTINUE
C
         E(L) = S * P
         D(L) = C * P
         IF (ABS(E(L)) .GT. B) GO TO 130
  210    P = D(L) + F
C     ********** ORDER EIGENVALUES **********
         IF (L .EQ. 1) GO TO 250
C     ********** FOR I=L STEP -1 UNTIL 2 DO -- **********
         DO 230 II = 2, L
            I = L + 2 - II
            IF (P .GE. D(I-1)) GO TO 270
            D(I) = D(I-1)
  230    CONTINUE
C
  250    I = 1
  270    D(I) = P
  290 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = L
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F290-2 TQL2

A Fortran IV Subroutine to Determine the Eigenvalues and Eigenvectors of a Symmetric Tridiagonal Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine TQL2 determines the eigenvalues and eigenvectors of a symmetric tridiagonal matrix. TQL2 uses the QL method to compute the eigenvalues and accumulates the QL transformations to compute the eigenvectors. The eigenvectors of a full symmetric matrix can also be computed directly by TQL2, if TRED2 (F278) has been used to reduce this matrix to tridiagonal form.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  TQL2(NM,N,D,E,Z,IERR)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array Z as specified in the DIMENSION statement for Z in the calling program.

N — is an integer input variable set equal to the order of the matrix. N must be not greater than NM.

D — is a working precision real one-dimensional variable of dimension at least N. On input, it contains the diagonal elements of the symmetric tridiagonal matrix. On output, it contains the eigenvalues of this matrix in ascending order.

E is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary. Note that TQL2 destroys E.

Z is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. If the eigenvectors of the symmetric tridiagonal matrix are desired, then on input, Z contains the identity matrix of order N, and on output, contains the orthonormal eigenvectors of this tridiagonal matrix. If the eigenvectors of a full symmetric matrix are desired, then on input, Z contains the transformation matrix produced in TRED2 which reduced the full matrix to tridiagonal form, and on output, contains the orthonormal eigenvectors of this full symmetric matrix.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

B. Error Conditions and Returns.

If more than 30 iterations are required to determine an eigenvalue, TQL2 terminates with IERR set to the index of the eigenvalue for which the failure occurs. The eigenvalues and eigenvectors in the D and Z arrays should be correct for indices 1,2,...,IERR-1, but the eigenvalues are unordered.

If all the eigenvalues are determined within 30 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine the eigenvalues and eigenvectors of a full symmetric matrix, TQL2 should be preceded by TRED2 (F278) to provide a suitable symmetric tridiagonal matrix for TQL2.

To determine the eigenvalues and eigenvectors of a complex Hermitian matrix, TQL2 should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for TQL2, and the input array Z to TQL2 should be initialized to the identity matrix. TQL2 should then be followed by HTRIBK (F285) to back transform the eigenvectors from TQL2 into those of the original matrix.

TQL2 does not perform well on matrices whose successive row sums vary widely in magnitude and are not strictly increasing from the first to the last row. The subroutine IMTQL2 (F292) is not sensitive to such row sums and is therefore recommended for symmetric tridiagonal matrices whose structure is not known.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the QL method. The essence of this method is a process whereby a sequence of symmetric tridiagonal matrices, unitarily similar to the original symmetric tridiagonal matrix, is formed which converges to a diagonal matrix. The rate of convergence of this sequence is improved by shifting the origin at each iteration. Before the iterations for each eigenvalue, the symmetric tridiagonal matrix is checked for a possible splitting into submatrices. If a splitting occurs, only the uppermost submatrix participates in the next iteration. The similarity transformations used in each iteration are accumulated in the Z array, producing the orthonormal eigenvectors for the original matrix. Finally, the eigenvalues are ordered in ascending order and the eigenvectors are ordered consistently.

The origin shift at each iteration is the eigenvalue of the current uppermost 2x2 principal minor closer to the first diagonal element of this minor. Whenever the uppermost 1x1 principal submatrix finally splits from the rest of the matrix, its element is taken to be an eigenvalue of the original matrix and the algorithm proceeds with the remaining submatrix. This process is continued until the matrix has split completely into submatrices of order 1. The tolerances in the splitting tests are proportional to the relative machine precision.

This subroutine is a translation of the Algol procedure TQL2 written and discussed in detail by Bowdler, Martin, Reinsch, and Wilkinson (1).

4. REFERENCES.

1) Bowdler, H., Martin, R.S., Reinsch, C., and Wilkinson, J.H., The QR and QL Algorithms for Symmetric Matrices, Num. Math. 11,293-306 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/3, 227-240, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine TQL2 is numerically stable (1); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE TQL2(NM,N,D,E,Z,IERR)
C
      INTEGER I,J,K,L,M,N,II,L1,NM,MML,IERR
      REAL D(N),E(N),Z(NM,N)
      REAL B,C,F,G,H,P,R,S,MACHEP
      REAL SQRT,ABS,SIGN
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      IF (N .EQ. 1) GO TO 1001
C
      DO 100 I = 2, N
  100 E(I-1) = E(I)
C
      F = 0.0
      B = 0.0
      E(N) = 0.0
C
      DO 240 L = 1, N
         J = 0
         H = MACHEP * (ABS(D(L)) + ABS(E(L)))
         IF (B .LT. H) B = H
C     ********** LOOK FOR SMALL SUB-DIAGONAL ELEMENT **********
         DO 110 M = L, N
            IF (ABS(E(M)) .LE. B) GO TO 120
C     ********** E(N) IS ALWAYS ZERO, SO THERE IS NO EXIT
C                THROUGH THE BOTTOM OF THE LOOP **********
  110    CONTINUE
C
  120    IF (M .EQ. L) GO TO 220
  130    IF (J .EQ. 30) GO TO 1000
         J = J + 1
C     ********** FORM SHIFT **********
         L1 = L + 1
         G = D(L)
         P = (D(L1) - G) / (2.0 * E(L))
         R = SQRT(P*P+1.0)
         D(L) = E(L) / (P + SIGN(R,P))
         H = G - D(L)
C
         DO 140 I = L1, N
  140    D(I) = D(I) - H
C
         F = F + H
C     ********** QL TRANSFORMATION **********
         P = D(M)
         C = 1.0
         S = 0.0
         MML = M - L
```

```
C     ********** FOR I=M-1 STEP -1 UNTIL L DO -- **********
         DO 200 II = 1, MML
            I = M - II
            G = C * E(I)
            H = C * P
            IF (ABS(P) .LT. ABS(E(I))) GO TO 150
            C = E(I) / P
            R = SQRT(C*C+1.0)
            E(I+1) = S * P * R
            S = C / R
            C = 1.0 / R
            GO TO 160
  150       C = P / E(I)
            R = SQRT(C*C+1.0)
            E(I+1) = S * E(I) * R
            S = 1.0 / R
            C = C * S
  160       P = C * D(I) - S * G
            D(I+1) = H + S * (C * G + S * D(I))
C     ********** FORM VECTOR **********
            DO 180 K = 1, N
               H = Z(K,I+1)
               Z(K,I+1) = S * Z(K,I) + C * H
               Z(K,I) = C * Z(K,I) - S * H
  180       CONTINUE
C
  200    CONTINUE
C
         E(L) = S * P
         D(L) = C * P
         IF (ABS(E(L)) .GT. B) GO TO 130
  220    D(L) = D(L) + F
  240 CONTINUE
C     ********** ORDER EIGENVALUES AND EIGENVECTORS **********
      DO 300 II = 2, N
         I = II - 1
         K = I
         P = D(I)
C
         DO 260 J = II, N
            IF (D(J) .GE. P) GO TO 260
            K = J
            P = D(J)
  260    CONTINUE
C
         IF (K .EQ. I) GO TO 300
         D(K) = D(I)
         D(I) = P
C
         DO 280 J = 1, N
            P = Z(J,I)
            Z(J,I) = Z(J,K)
            Z(J,K) = P
  280    CONTINUE
```

```
C
  300 CONTINUE
C
      GO TO 1001
C     ********** SET ERROR -- NO CONVERGENCE TO AN
C                EIGENVALUE AFTER 30 ITERATIONS **********
 1000 IERR = L
 1001 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F279-2 TRBAK1

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Symmetric Tridiagonal Matrix Determined by TRED1.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine TRBAK1 forms the eigenvectors of a real symmetric matrix from the eigenvectors of that symmetric tridiagonal matrix determined by TRED1 (F277).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE TRBAK1(NM,N,A,E,M,Z)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

N — is an integer input variable set equal to the order of the matrix. N must be not greater than NM.

A — is a working precision real input two-dimensional variable with row dimension NM and column dimension at least N. The strict lower triangle of A contains some information about the orthogonal transformations used in the reduction to the tridiagonal form. The remaining upper part of the matrix is arbitrary. See section 3 of F277 for the details.

E is a working precision real input one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the tridiagonal matrix. The element E(1) is arbitrary. These elements serve to provide the remaining information about the orthogonal transformations.

M is an integer input variable set equal to the number of columns of Z to be back transformed.

Z is a working precision real two-dimensional variable with row dimension NM and column dimension at least M. On input, the first M columns of Z contain the eigenvectors to be back transformed. On output, these columns of Z contain the transformed eigenvectors. The transformed eigenvectors are orthonormal if the input eigenvectors are orthonormal.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine TRED1 (F277).

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the symmetric matrix C (say) has been reduced to the tridiagonal symmetric matrix F by the similarity transformation

$$F = Q^T C Q$$

where Q is a product of the orthogonal symmetric matrices encoded in E and in the strict lower triangle of A. Then, given an array Z of column vectors, TRBAK1 computes the matrix product QZ. If the eigenvectors of F are columns of the array Z, then TRBAK1 forms the eigenvectors of C in their place. Since Q is orthogonal, vector Euclidean norms are preserved.

This subroutine is a translation of the Algol procedure TRBAK1 written and discussed in detail by Martin, Reinsch, and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S., Reinsch, C., and Wilkinson, J.H., Householder's Tridiagonalization of a Symmetric Matrix, Num. Math. 11,181-195 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/2, 212-226, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real symmetric matrices.

B. Accuracy.

The accuracy of TRBAK1 can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real symmetric matrices and matrix systems. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix or system close to the original matrix or system and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix or system.

```
      SUBROUTINE TRBAK1(NM,N,A,E,M,Z)
C
      INTEGER I,J,K,L,M,N,NM
      REAL A(NM,N),E(N),Z(NM,M)
      REAL S
C
      IF (M .EQ. 0) GO TO 200
      IF (N .EQ. 1) GO TO 200
C
      DO 140 I = 2, N
         L = I - 1
         IF (E(I) .EQ. 0.0) GO TO 140
C
         DO 130 J = 1, M
            S = 0.0
C
            DO 110 K = 1, L
  110       S = S + A(I,K) * Z(K,J)
C     ********** DIVISOR BELOW IS NEGATIVE OF H FORMED IN TRED1.
C                DOUBLE DIVISION AVOIDS POSSIBLE UNDERFLOW **********
            S = (S / A(I,L)) / E(I)
C
            DO 120 K = 1, L
  120       Z(K,J) = Z(K,J) + S * A(I,K)
C
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F229 TRBAK3

A Fortran IV Subroutine to Back Transform the Eigenvectors of that Symmetric Tridiagonal Matrix Determined by TRED3.

July, 1975

1. PURPOSE.

The Fortran IV subroutine TRBAK3 forms the eigenvectors of a real symmetric matrix from the eigenvectors of that symmetric tridiagonal matrix determined by TRED3 (F228).

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE TRBAK3(NM,N,NV,A,M,Z)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array Z as specified in the DIMENSION statement for Z in the calling program.

N — is an integer input variable set equal to the order of the matrix. N must be not greater than NM.

NV — is an integer input variable set equal to the dimension of the array A as specified in the DIMENSION statement for A in the calling program. NV must be not less than N*(N+1)/2.

A — is a working precision real input one-dimensional variable of dimension at least N*(N+1)/2 containing information about the orthogonal transformations used in the reduction to the tridiagonal form. See section 3 of F228 for the details.

M is an integer input variable set equal to the number of columns of Z to be back transformed.

Z is a working precision real two-dimensional variable with row dimension NM and column dimension at least M. On input, the first M columns of Z contain the eigenvectors to be back transformed. On output, these columns of Z contain the transformed eigenvectors. The transformed eigenvectors are orthonormal if the input eigenvectors are orthonormal.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

This subroutine should be used in conjunction with the subroutine TRED3 (F228).

3. DISCUSSION OF METHOD AND ALGORITHM.

Suppose that the symmetric matrix C (say) has been reduced to the tridiagonal symmetric matrix F by the similarity transformation

$$F = Q^T C Q$$

where Q is a product of the orthogonal symmetric matrices encoded in A. Then, given an array Z of column vectors, TRBAK3 computes the matrix product QZ. If the eigenvectors of F are columns of the array Z, then TRBAK3 forms the eigenvectors of C in their place. Since Q is orthogonal, vector Euclidean norms are preserved.

This subroutine is a translation of the Algol procedure TRBAK3 written and discussed in detail by Martin, Reinsch, and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S., Reinsch, C., and Wilkinson, J.H., Householder's Tridiagonalization of a Symmetric Matrix, Num. Math. 11,181-195 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/2, 212-226, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real symmetric packed matrices.

B. Accuracy.

The accuracy of TRBAK3 can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real symmetric packed matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE TRBAK3(NM,N,NV,A,M,Z)
C
      INTEGER I,J,K,L,M,N,IK,IZ,NM,NV
      REAL A(NV),Z(NM,M)
      REAL H,S
C
      IF (M .EQ. 0) GO TO 200
      IF (N .EQ. 1) GO TO 200
C
      DO 140 I = 2, N
         L = I - 1
         IZ = (I * L) / 2
         IK = IZ + I
         H = A(IK)
         IF (H .EQ. 0.0) GO TO 140
C
         DO 130 J = 1, M
            S = 0.0
            IK = IZ
C
            DO 110 K = 1, L
               IK = IK + 1
               S = S + A(IK) * Z(K,J)
  110       CONTINUE
C     ********** DOUBLE DIVISION AVOIDS POSSIBLE UNDERFLOW **********
            S = (S / H) / H
            IK = IZ
C
            DO 120 K = 1, L
               IK = IK + 1
               Z(K,J) = Z(K,J) - S * A(IK)
  120       CONTINUE
C
  130    CONTINUE
C
  140 CONTINUE
C
  200 RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F277-2 TRED1

A Fortran IV Subroutine to Reduce a Real Symmetric Matrix to a Symmetric Tridiagonal Matrix Using Orthogonal Transformations.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine TRED1 reduces a real symmetric matrix to a symmetric tridiagonal matrix using orthogonal similarity transformations. This reduced form is used by other subroutines to find the eigenvalues and/or eigenvectors of the original matrix. See section 2C for the specific routines.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  TRED1(NM,N,A,D,E,E2)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array A as specified in the DIMENSION statement for A in the calling program.

N — is an integer input variable set equal to the order of the matrix A. N must be not greater than NM.

A — is a working precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, A contains the symmetric matrix of order N to be reduced to tridiagonal form. Only the full lower triangle of the matrix need be

supplied. On output, the strict lower triangle of A contains information about the orthogonal transformations used in the reduction. The full upper triangle of A is unaltered. See section 3 for the details.

D — is a working precision real output one-dimensional variable of dimension at least N containing the diagonal elements of the tridiagonal matrix.

E — is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the tridiagonal matrix. The element E(1) is set to zero.

E2 — is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the squares of the subdiagonal elements of the tridiagonal matrix. The element E2(1) is set to zero. E2 need not be distinct from E (non-standard usage acceptable with at least those compilers included in the certification statement), in which case no squares are returned.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues of the original matrix are desired, this subroutine should be followed by TQL1 (F289), IMTQL1 (F291), or TQLRAT (F235).

If some of the eigenvalues of the original matrix are desired, this subroutine should be followed by BISECT (F294) or TRIDIB (F237).

If some of the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by TSTURM (F293), or by BISECT (F294) and TINVIT (F223), or by TRIDIB (F237) and TINVIT, or by IMTQLV (F234) and TINVIT, and then by TRBAK1 (F279).

If all the eigenvalues and eigenvectors of the original matrix are desired, subroutine TRED2 (F278) should be used rather than TRED1 to perform the tridiagonal reduction, and should be followed by TQL2 (F290) or IMTQL2 (F292).

If the matrix has elements of widely varying magnitudes, the smaller ones should be in the top left-hand corner.

3. DISCUSSION OF METHOD AND ALGORITHM.

The tridiagonal reduction is performed in the following way. Starting with J=N, the elements in the J-th row to the left of the diagonal are first scaled, to avoid possible underflow in the transformation that might result in severe departure from orthogonality. The sum of squares SIGMA of these scaled elements is next formed. Then, a vector U and a scalar

$$H = U^T U/2$$

define an operator

$$P = I - UU^T/H$$

which is orthogonal and symmetric and for which the similarity transformation PAP eliminates the elements in the J-th row of A to the left of the subdiagonal and the symmetrical elements in the J-th column.

The non-zero components of U are the elements of the J-th row to the left of the diagonal with the last of them augmented by the square root of SIGMA prefixed by the sign of the subdiagonal element. By storing the transformed subdiagonal element in E(J) and not overwriting the row elements eliminated in the transformation, full information about P is saved for later use in TRBAK1.

The transformation sets E2(J) equal to SIGMA and E(J) equal to the square root of SIGMA prefixed by sign opposite to that of the replaced subdiagonal element.

The above steps are repeated on further rows of the transformed A in reverse order until A is reduced to tridiagonal form; that is, repeated for J = N-1,N-2,...,3.

Only the elements in the lower triangle of A are accessed, and although the diagonal elements are modified in the algorithm, they are restored to their original contents by the end of the subroutine, thus preserving the full upper triangle of A.

This subroutine is a translation of the Algol procedure TRED1 written and discussed in detail by Martin, Reinsch, and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S., Reinsch, C., and Wilkinson, J.H., Householder's Tridiagonalization of a Symmetric Matrix, Num. Math. 11,181-195 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/2, 212-226, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real symmetric matrices.

B. Accuracy.

The accuracy of TRED1 can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real symmetric matrices and matrix systems. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix or system close to the original matrix or system and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix or system.

```
      SUBROUTINE TRED1(NM,N,A,D,E,E2)
C
      INTEGER I,J,K,L,N,II,NM,JP1
      REAL A(NM,N),D(N),E(N),E2(N)
      REAL F,G,H,SCALE
      REAL SQRT,ABS,SIGN
C
      DO 100 I = 1, N
  100 D(I) = A(I,I)
C     ********** FOR I=N STEP -1 UNTIL 1 DO -- **********
      DO 300 II = 1, N
         I = N + 1 - II
         L = I - 1
         H = 0.0
         SCALE = 0.0
         IF (L .LT. 1) GO TO 130
C     ********** SCALE ROW (ALGOL TOL THEN NOT NEEDED) **********
         DO 120 K = 1, L
  120    SCALE = SCALE + ABS(A(I,K))
C
         IF (SCALE .NE. 0.0) GO TO 140
  130    E(I) = 0.0
         E2(I) = 0.0
         GO TO 290
C
  140    DO 150 K = 1, L
            A(I,K) = A(I,K) / SCALE
            H = H + A(I,K) * A(I,K)
  150    CONTINUE
C
         E2(I) = SCALE * SCALE * H
         F = A(I,L)
         G = -SIGN(SQRT(H),F)
         E(I) = SCALE * G
         H = H - F * G
         A(I,L) = F - G
         IF (L .EQ. 1) GO TO 270
         F = 0.0
C
         DO 240 J = 1, L
            G = 0.0
C     ********** FORM ELEMENT OF A*U **********
            DO 180 K = 1, J
  180       G = G + A(J,K) * A(I,K)
C
            JP1 = J + 1
            IF (L .LT. JP1) GO TO 220
C
            DO 200 K = JP1, L
  200       G = G + A(K,J) * A(I,K)
C     ********** FORM ELEMENT OF P **********
  220       E(J) = G / H
            F = F + E(J) * A(I,J)
  240    CONTINUE
```

```
C
         H = F / (H + H)
C      ********** FORM REDUCED A **********
         DO 260 J = 1, L
            F = A(I,J)
            G = E(J) - H * F
            E(J) = G
C
            DO 260 K = 1, J
               A(J,K) = A(J,K) - F * E(K) - G * A(I,K)
  260    CONTINUE
C
  270    DO 280 K = 1, L
  280    A(I,K) = SCALE * A(I,K)
C
  290    H = D(I)
         D(I) = A(I,I)
         A(I,I) = H
  300 CONTINUE
C
       RETURN
       END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F278-2 TRED2

A Fortran IV Subroutine to Reduce a Real Symmetric Matrix to a Symmetric Tridiagonal Matrix Accumulating the Orthogonal Transformations.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine TRED2 reduces a real symmetric matrix to a symmetric tridiagonal matrix using and accumulating orthogonal similarity transformations. This reduced form and the transformation matrix are used by subroutine TQL2 (F290) or IMTQL2 (F292) to find the eigenvalues and eigenvectors of the original matrix.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  TRED2(NM,N,A,D,E,Z)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional arrays A and Z as specified in the DIMENSION statements for A and Z in the calling program.

N — is an integer input variable set equal to the order of the matrix A. N must be not greater than NM.

A is a working precision real input two-dimensional variable with row dimension NM and column dimension at least N. A contains the symmetric matrix of order N to be reduced to tridiagonal form. Only the full lower triangle of the matrix need be supplied.

D is a working precision real output one-dimensional variable of dimension at least N containing the diagonal elements of the tridiagonal matrix.

E is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the tridiagonal matrix. The element E(1) is set to zero.

Z is a working precision real output two-dimensional variable with row dimension NM and column dimension at least N. It contains the orthogonal transformation matrix produced in the reduction to the tridiagonal form.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by TQL2 (F290) or IMTQL2 (F292).

If some other combination of eigenvalues and eigenvectors is desired, subroutine TRED1 (F277) should be used rather than TRED2 to perform the tridiagonal reduction.

If the matrix has elements of widely varying magnitudes, the smaller ones should be in the top left-hand corner.

Parameters A and Z need not be distinct.

3. DISCUSSION OF METHOD AND ALGORITHM.

The lower triangle of A is initially copied into Z and all subsequent operations are performed on Z.

The tridiagonal reduction is performed in the following way. Starting with J=N, the elements in the J-th row to the left of the diagonal are first scaled, to avoid possible underflow in the transformation that might result in severe departure from orthogonality. The sum of squares SIGMA of these scaled elements is next formed. Then, a vector U and a scalar

$$H = U^T U/2$$

define an operator

$$P = I - UU^T/H$$

which is orthogonal and symmetric and for which the similarity transformation PAP eliminates the elements in the J-th row of A to the left of the subdiagonal and the symmetrical elements in the J-th column.

The non-zero components of U are the elements of the J-th row to the left of the diagonal with the last of them augmented by the square root of SIGMA prefixed by the sign of the subdiagonal element. By storing the transformed subdiagonal element in E(J) and not overwriting the row elements eliminated in the transformation, full information about P is saved for later accumulation of transformations.

The transformation sets E(J) equal to the square root of SIGMA prefixed by sign opposite to that of the replaced subdiagonal element.

The above steps are repeated on further rows of the transformed A in reverse order until A is reduced to tridiagonal form; that is, repeated for J = N-1,N-2,...,3.

Finally, the orthogonal transformation matrix is accumulated in Z as the product of the N-2 operators defined in the tridiagonal reduction.

This subroutine is a translation of the Algol procedure TRED2 written and discussed in detail by Martin, Reinsch, and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S., Reinsch, C., and Wilkinson, J.H., Householder's Tridiagonalization of a Symmetric Matrix, Num. Math. 11,181-195 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/2, 212-226, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real symmetric matrices.

B. Accuracy.

The accuracy of TRED2 can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real symmetric matrices and matrix systems. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix or system close to the original matrix or system and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix or system.

```
      SUBROUTINE TRED2(NM,N,A,D,E,Z)
C
      INTEGER I,J,K,L,N,II,NM,JP1
      REAL A(NM,N),D(N),E(N),Z(NM,N)
      REAL F,G,H,HH,SCALE
      REAL SQRT,ABS,SIGN
C
      DO 100 I = 1, N
C
         DO 100 J = 1, I
            Z(I,J) = A(I,J)
  100 CONTINUE
C
      IF (N .EQ. 1) GO TO 320
C     ********** FOR I=N STEP -1 UNTIL 2 DO -- **********
      DO 300 II = 2, N
         I = N + 2 - II
         L = I - 1
         H = 0.0
         SCALE = 0.0
         IF (L .LT. 2) GO TO 130
C     ********** SCALE ROW (ALGOL TOL THEN NOT NEEDED) **********
         DO 120 K = 1, L
  120    SCALE = SCALE + ABS(Z(I,K))
C
         IF (SCALE .NE. 0.0) GO TO 140
  130    E(I) = Z(I,L)
         GO TO 290
C
  140    DO 150 K = 1, L
            Z(I,K) = Z(I,K) / SCALE
            H = H + Z(I,K) * Z(I,K)
  150    CONTINUE
C
         F = Z(I,L)
         G = -SIGN(SQRT(H),F)
         E(I) = SCALE * G
         H = H - F * G
         Z(I,L) = F - G
         F = 0.0
C
         DO 240 J = 1, L
            Z(J,I) = Z(I,J) / H
            G = 0.0
C     ********** FORM ELEMENT OF A*U **********
            DO 180 K = 1, J
  180       G = G + Z(J,K) * Z(I,K)
C
            JP1 = J + 1
            IF (L .LT. JP1) GO TO 220
C
            DO 200 K = JP1, L
  200       G = G + Z(K,J) * Z(I,K)
```

```
C     ********** FORM ELEMENT OF P **********
  220       E(J) = G / H
            F = F + E(J) * Z(I,J)
  240    CONTINUE
C
         HH = F / (H + H)
C     ********** FORM REDUCED A **********
         DO 260 J = 1, L
            F = Z(I,J)
            G = E(J) - HH * F
            E(J) = G
C
            DO 260 K = 1, J
               Z(J,K) = Z(J,K) - F * E(K) - G * Z(I,K)
  260    CONTINUE
C
  290    D(I) = H
  300 CONTINUE
C
  320 D(1) = 0.0
      E(1) = 0.0
C     ********** ACCUMULATION OF TRANSFORMATION MATRICES **********
      DO 500 I = 1, N
         L = I - 1
         IF (D(I) .EQ. 0.0) GO TO 380
C
         DO 360 J = 1, L
            G = 0.0
C
            DO 340 K = 1, L
  340       G = G + Z(I,K) * Z(K,J)
C
            DO 360 K = 1, L
               Z(K,J) = Z(K,J) - G * Z(K,I)
  360    CONTINUE
C
  380    D(I) = Z(I,I)
         Z(I,I) = 1.0
         IF (L .LT. 1) GO TO 500
C
         DO 400 J = 1, L
            Z(I,J) = 0.0
            Z(J,I) = 0.0
  400    CONTINUE
C
  500 CONTINUE
C
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F228 TRED3

A Fortran IV Subroutine to Reduce a Real Symmetric Matrix, Stored as a One-Dimensional Array, to a Symmetric Tridiagonal Matrix Using Orthogonal Transformations.

July, 1975

1. PURPOSE.

The Fortran IV subroutine TRED3 reduces a real symmetric matrix, stored as a one-dimensional array, to a symmetric tridiagonal matrix using orthogonal similarity transformations. This reduced form is used by other subroutines to find the eigenvalues and/or eigenvectors of the original matrix. See section 2C for the specific routines.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  TRED3(N,NV,A,D,E,E2)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

N — is an integer input variable set equal to the order of the matrix A.

NV — is an integer input variable set equal to the dimension of the array A as specified in the DIMENSION statement for A in the calling program. NV must be not less than N*(N+1)/2.

A — is a working precision real one-dimensional variable of dimension at least N*(N+1)/2. On input, A contains the lower triangle of the symmetric matrix of order N to be reduced to tridiagonal form, packed row-wise. For example if N=3, A should contain

(A(1,1),A(2,1),A(2,2),A(3,1),A(3,2),A(3,3))

where the subscripts for each element refer to the row and column of the element in the standard two-dimensional representation. On output, A contains information about the orthogonal transformations used in the reduction. See section 3 for the details.

D is a working precision real output one-dimensional variable of dimension at least N containing the diagonal elements of the tridiagonal matrix.

E is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the tridiagonal matrix. The element E(1) is set to zero.

E2 is a working precision real output one-dimensional variable of dimension at least N containing, in its last N-1 positions, the squares of the subdiagonal elements of the tridiagonal matrix. The element E2(1) is set to zero. E2 need not be distinct from E (non-standard usage acceptable with at least those compilers included in the certification statement), in which case no squares are returned.

B. Error Conditions and Returns.

None.

C. Applicability and Restrictions.

If all the eigenvalues of the original matrix are desired, this subroutine should be followed by TQL1 (F289), IMTQL1 (F291), or TQLRAT (F235).

If some of the eigenvalues of the original matrix are desired, this subroutine should be followed by BISECT (F294) or TRIDIB (F237).

If some of the eigenvalues and eigenvectors of the original matrix are desired, this subroutine should be followed by TSTURM (F293), or by BISECT (F294) and TINVIT (F223), or by TRIDIB (F237) and TINVIT, or by IMTQLV (F234) and TINVIT, and then by TRBAK3 (F229).

If all the eigenvalues and eigenvectors of the original matrix are desired, subroutine TRED2 (F278) should be used rather than TRED3 to perform the tridiagonal reduction, and should be followed by TQL2 (F290) or IMTQL2 (F292). In this case, the packed form of the matrix cannot be used, but the eigenvectors can share the space required for the matrix.

If the matrix has elements of widely varying magnitudes, the smaller ones should be at the top.

3. DISCUSSION OF METHOD AND ALGORITHM.

Discussion of the algorithm is facilitated if the matrix is considered square. The implementation, however, achieves significant storage economy by specifying only the lower triangle to be stored and packed one-dimensionally.

The tridiagonal reduction is performed in the following way. Starting with J=N, the elements in the J-th row to the left of the diagonal are first scaled, to avoid possible underflow in the transformation that might result in severe departure from orthogonality. The sum of squares SIGMA of these scaled elements is next formed. Then, a vector U and a scalar

$$H = U\,U^T/2$$

define an operator

$$P = I - UU^T/H$$

which is orthogonal and symmetric and for which the similarity transformation PAP eliminates the elements in the J-th row of A to the left of the subdiagonal and the symmetrical elements in the J-th column.

The non-zero components of U are the elements of the J-th row to the left of the diagonal with the last of them augmented by the square root of SIGMA prefixed by the sign of the subdiagonal element. By storing the transformed subdiagonal element in E(J) and not overwriting the row elements eliminated in the transformation, full information about P is saved for later use in TRBAK3.

The transformation sets E2(J) equal to SIGMA and E(J) equal to the square root of SIGMA prefixed by sign opposite to that of the replaced subdiagonal element.

The above steps are repeated on further rows of the transformed A in reverse order until A is reduced to tridiagonal form; that is, repeated for J = N-1,N-2,...,3.

This subroutine is a translation of the Algol procedure TRED3 written and discussed in detail by Martin, Reinsch, and Wilkinson (1).

4. REFERENCES.

1) Martin, R.S., Reinsch, C., and Wilkinson, J.H., Householder's Tridiagonalization of a Symmetric Matrix, Num. Math. 11,181-195 (1968). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/2, 212-226, Springer-Verlag, 1971.)

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for real symmetric packed matrices.

B. Accuracy.

The accuracy of TRED3 can best be described in terms of its role in those paths of EISPACK which find eigenvalues and eigenvectors of real symmetric packed matrices. In these paths, this subroutine is numerically stable (1). This stability contributes to the property of these paths that the computed eigenvalues are the exact eigenvalues of a matrix close to the original matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE TRED3(N,NV,A,D,E,E2)
C
      INTEGER I,J,K,L,N,II,IZ,JK,NV
      REAL A(NV),D(N),E(N),E2(N)
      REAL F,G,H,HH,SCALE
      REAL SQRT,ABS,SIGN
C
C     ********** FOR I=N STEP -1 UNTIL 1 DO -- **********
      DO  300 II = 1, N
         I = N + 1 - II
         L = I - 1
         IZ = (I * L) / 2
         H = 0.0
         SCALE = 0.0
         IF (L .LT. 1) GO TO 130
C     ********** SCALE ROW (ALGOL TOL THEN NOT NEEDED) **********
         DO 120 K = 1, L
            IZ = IZ + 1
            D(K) = A(IZ)
            SCALE = SCALE + ABS(D(K))
  120    CONTINUE
C
         IF (SCALE .NE. 0.0) GO TO 140
  130    E(I) = 0.0
         E2(I) = 0.0
         GO TO 290
C
  140    DO 150 K = 1, L
            D(K) = D(K) / SCALE
            H = H + D(K) * D(K)
  150    CONTINUE
C
         E2(I) = SCALE * SCALE * H
         F = D(L)
         G = -SIGN(SQRT(H),F)
         E(I) = SCALE * G
         H = H - F * G
         D(L) = F - G
         A(IZ) = SCALE * D(L)
         IF (L .EQ. 1) GO TO 290
         F = 0.0
C
         DO 240 J = 1, L
            G = 0.0
            JK = (J * (J-1)) / 2
C     ********** FORM ELEMENT OF A*U **********
            DO 180 K = 1, L
               JK = JK + 1
               IF (K .GT. J) JK = JK + K - 2
               G = G + A(JK) * D(K)
  180       CONTINUE
```

```
C      ********** FORM ELEMENT OF P **********
               E(J) = G / H
               F = F + E(J) * D(J)
  240     CONTINUE
C
          HH = F / (H + H)
          JK = 0
C      ********** FORM REDUCED A **********
          DO 260 J = 1, L
             F = D(J)
             G = E(J) - HH * F
             E(J) = G
C
             DO 260 K = 1, J
                JK = JK + 1
                A(JK) = A(JK) - F * E(K) - G * D(K)
  260     CONTINUE
C
  290     D(I) = A(IZ+1)
          A(IZ+1) = SCALE * SQRT(H)
  300 CONTINUE
C
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F237 TRIDIB

A Fortran IV Subroutine to Determine Some Eigenvalues of a Symmetric Tridiagonal Matrix.

July, 1975

1. PURPOSE.

The Fortran IV subroutine TRIDIB determines those eigenvalues of a symmetric tridiagonal matrix between specified boundary indices using Sturm sequencing.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  TRIDIB(N,EPS1,D,E,E2,LB,UB,
                   M11,M,W,IND,IERR,RV4,RV5)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

N — is an integer input variable set equal to the order of the matrix.

EPS1 — is a working precision real variable. On input, it specifies an absolute error tolerance for the computed eigenvalues. If the input EPS1 is non-positive, it is reset to a default value described in section 2C.

D — is a working precision real input one-dimensional variable of dimension at least N containing the diagonal elements of the symmetric tridiagonal matrix.

E — is a working precision real input one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary.

E2 is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the squares of the subdiagonal elements of the symmetric tridiagonal matrix. E2(1) is arbitrary. On output, E2(1) is set to zero. If any of the elements in E are regarded as negligible, the corresponding elements of E2 are set to zero, and so the matrix splits into a direct sum of submatrices.

LB,UB are working precision real output variables set to the lower and upper endpoints, respectively, of an interval containing exactly the desired eigenvalues. See section 2C for further details.

M11 is an integer input variable specifying the lower index of the set of desired (smallest) eigenvalues.

M is an integer input variable specifying the number of eigenvalues desired. The upper index, M22, of the set is then obtained internally as M22=M11+M-1.

W is a working precision real output one-dimensional variable of dimension at least M containing the M eigenvalues of the symmetric tridiagonal matrix between boundary indices M11 and M22. The eigenvalues are in ascending order in W.

IND is an integer output one-dimensional variable of dimension at least M containing the submatrix indices associated with the corresponding M eigenvalues in W. Eigenvalues belonging to the first submatrix have index 1, those belonging to the second submatrix have index 2, etc.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

RV4,RV5 are working precision real temporary one-dimensional variables of dimension at least N used to hold the lower and upper bounds for the eigenvalues in the bisection process.

B. Error Conditions and Returns.

If exactly multiple eigenvalues prevent the determination of an appropriate LB, TRIDIB terminates with no eigenvalues computed, and IERR is set to 3*N+1. In this case, the returned values of LB and UB define the Gerschgorin interval containing all the eigenvalues of the matrix.

If, after determining LB, exactly multiple eigenvalues prevent the determination of an appropriate UB, TRIDIB terminates with no eigenvalues computed, and IERR is set to 3*N+2. In this case, the returned value of UB is the upper bound for the Gerschgorin interval containing all the eigenvalues of the matrix.

If neither of the above error conditions occurs, IERR is set to zero.

C. Applicability and Restrictions.

To determine some of the eigenvalues of a full symmetric matrix, TRIDIB should be preceded by TRED1 (F277) to provide a suitable symmetric tridiagonal matrix for TRIDIB.

To determine some of the eigenvalues of a complex Hermitian matrix, TRIDIB should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for TRIDIB.

Some of the eigenvalues of certain non-symmetric tridiagonal matrices can be computed using the combination of FIGI (F280) and TRIDIB. See F280 for the description of this special class of matrices. For these matrices, TRIDIB should be preceded by FIGI to provide a suitable symmetric matrix for TRIDIB.

To determine eigenvectors associated with the computed eigenvalues, TRIDIB should be followed by TINVIT (F223) and the appropriate back transformation subroutine -- TRBAK1 (F279) after TRED1, HTRIBK (F285) after HTRIDI, or BAKVEC (F281) after FIGI.

The subroutines TQL1 (F289), IMTQL1 (F291), and TQLRAT (F235) determine all the eigenvalues of a symmetric tridiagonal matrix faster than TRIDIB determines 25 percent of them. Hence, if more than 25 percent of them are desired, it is recommended that TQL1, IMTQL1, or TQLRAT be used.

If it is preferred to specify the interval (LB,UB) to be searched for eigenvalues rather than the boundary indices M11 and M22, use subroutine BISECT (F294) instead.

The precision of the computed eigenvalues is controlled through the parameter EPS1. To obtain eigenvalues accurate to within a certain absolute error, EPS1 should be set to that error. In particular, if MACHEP denotes the relative machine precision, then to obtain the eigenvalues to an accuracy commensurate with small relative perturbations of the order of MACHEP in the matrix elements, it is enough for most tridiagonal matrices that EPS1 be approximately MACHEP times a norm of the matrix. But some matrices require a smaller EPS1 for this accuracy, perhaps as small as MACHEP times the eigenvalue of smallest magnitude in (LB,UB). Note, however, that if EPS1 is smaller than required, TRIDIB will perform unnecessary bisection steps to determine the eigenvalues. For further discussion of EPS1, see reference (1).

If the input EPS1 is non-positive, TRIDIB resets it, for each submatrix, to -MACHEP times the 1-norm of the submatrix and uses its magnitude. This value is tentatively considered adequate for computing the eigenvalues to an accuracy commensurate with small relative perturbations of the order of MACHEP in the matrix elements.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the method of bisection applied to the Sturm sequence.

The calculations proceed as follows. First, the subdiagonal elements are tested for negligibility. If an element is considered negligible, its square is set to zero, and so the matrix splits into a direct sum of submatrices. At the same time, the Gerschgorin interval known to contain all the eigenvalues of the matrix is determined. Then, an interval (LB,UB) is determined, using a bisection process proceeding from the Gerschgorin bounds, that contains exactly eigenvalues M11 through M22.

Next, a submatrix is examined for its eigenvalues in the interval (LB,UB). Its Gerschgorin interval is determined and used to refine the interval (LB,UB). If the input EPS1 is non-positive, it is reset to the default value described in section 2C. Then, subintervals, each enclosing an eigenvalue in (LB,UB), are shrunk using a bisection process

until the endpoints of each subinterval are close enough to be accepted as an eigenvalue of the submatrix. Here the endpoints of each subinterval are close enough when they differ by less than MACHEP times twice the sum of the magnitudes of the endpoints plus the absolute error tolerance EPS1.

The submatrix eigenvalues are then merged with previously found eigenvalues into an ordered set.

The above steps are repeated on each submatrix until all the eigenvalues between boundary indices M11 and M22 are computed.

This subroutine is a translation of the Algol procedure BISECT written and discussed in detail by Barth, Martin, and Wilkinson (1). A similar Algol procedure TRISTURM is discussed in detail by Peters and Wilkinson (2); a subset of it has been translated as subroutine BISECT (F294).

4. REFERENCES.

1) Barth, W., Martin, R.S., and Wilkinson, J.H., Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection, Num. Math. 9,386-393 (1967). (Reprinted in Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/5, 249-256, Springer-Verlag, 1971.)

2) Peters, G. and Wilkinson, J.H., The Calculation of Specified Eigenvectors by Inverse Iteration, Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/18, 418-439, Springer-Verlag, 1971.

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine TRIDIB is numerically stable (1,2); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix.

```
      SUBROUTINE TRIDIB(N,EPS1,D,E,E2,LB,UB,M11,M,W,IND,IERR,RV4,RV5)
C
      INTEGER I,J,K,L,M,N,P,Q,R,S,II,M1,M2,M11,M22,TAG,IERR,ISTURM
      REAL D(N),E(N),E2(N),W(M),RV4(N),RV5(N)
      REAL U,V,LB,T1,T2,UB,XU,X0,X1,EPS1,MACHEP
      REAL ABS,AMAX1,AMIN1,FLOAT
      INTEGER IND(M)
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      TAG = 0
      XU = D(1)
      X0 = D(1)
      U = 0.0
C     ********** LOOK FOR SMALL SUB-DIAGONAL ENTRIES AND DETERMINE AN
C                INTERVAL CONTAINING ALL THE EIGENVALUES **********
      DO 40 I = 1, N
         X1 = U
         U = 0.0
         IF (I .NE. N) U = ABS(E(I+1))
         XU = AMIN1(D(I)-(X1+U),XU)
         X0 = AMAX1(D(I)+(X1+U),X0)
         IF (I .EQ. 1) GO TO 20
         IF (ABS(E(I)) .GT. MACHEP * (ABS(D(I)) + ABS(D(I-1))))
     X      GO TO 40
   20    E2(I) = 0.0
   40 CONTINUE
C
      X1 = AMAX1(ABS(XU),ABS(X0)) * MACHEP * FLOAT(N)
      XU = XU - X1
      T1 = XU
      X0 = X0 + X1
      T2 = X0
C     ********** DETERMINE AN INTERVAL CONTAINING EXACTLY
C                THE DESIRED EIGENVALUES **********
      P = 1
      Q = N
      M1 = M11 - 1
      IF (M1 .EQ. 0) GO TO 75
      ISTURM = 1
   50 V = X1
      X1 = (XU + X0) * 0.5
      IF (X1 .EQ. V) GO TO 980
      GO TO 320
   60 IF (S - M1) 65, 73, 70
   65 XU = X1
      GO TO 50
   70 X0 = X1
      GO TO 50
```

```
   73 XU = X1
      T1 = X1
   75 M22 = M1 + M
      IF (M22 .EQ. N) GO TO 90
      X0 = T2
      ISTURM = 2
      GO TO 50
   80 IF (S - M22) 65, 85, 70
   85 T2 = X1
   90 Q = 0
      R = 0
C     ********** ESTABLISH AND PROCESS NEXT SUBMATRIX, REFINING
C                INTERVAL BY THE GERSCHGORIN BOUNDS **********
  100 IF (R .EQ. M) GO TO 1001
      TAG = TAG + 1
      P = Q + 1
      XU = D(P)
      X0 = D(P)
      U = 0.0
C
      DO 120 Q = P, N
         X1 = U
         U = 0.0
         V = 0.0
         IF (Q .EQ. N) GO TO 110
         U = ABS(E(Q+1))
         V = E2(Q+1)
  110    XU = AMIN1(D(Q)-(X1+U),XU)
         X0 = AMAX1(D(Q)+(X1+U),X0)
         IF (V .EQ. 0.0) GO TO 140
  120 CONTINUE
C
  140 X1 = AMAX1(ABS(XU),ABS(X0)) * MACHEP
      IF (EPS1 .LE. 0.0) EPS1 = -X1
      IF (P .NE. Q) GO TO 180
C     ********** CHECK FOR ISOLATED ROOT WITHIN INTERVAL **********
      IF (T1 .GT. D(P) .OR. D(P) .GE. T2) GO TO 940
      M1 = P
      M2 = P
      RV5(P) = D(P)
      GO TO 900
  180 X1 = X1 * FLOAT(Q-P+1)
      LB = AMAX1(T1,XU-X1)
      UB = AMIN1(T2,X0+X1)
      X1 = LB
      ISTURM = 3
      GO TO 320
  200 M1 = S + 1
      X1 = UB
      ISTURM = 4
      GO TO 320
  220 M2 = S
      IF (M1 .GT. M2) GO TO 940
```

```
C     ********** FIND ROOTS BY BISECTION **********
      X0 = UB
      ISTURM = 5
C
      DO 240 I = M1, M2
         RV5(I) = UB
         RV4(I) = LB
  240 CONTINUE
C     ********** LOOP FOR K-TH EIGENVALUE
C                FOR K=M2 STEP -1 UNTIL M1 DO --
C                (-DO- NOT USED TO LEGALIZE COMPUTEE-GO-TO) **********
      K = M2
  250    XU = LB
C     ********** FOR I=K STEP -1 UNTIL M1 DO -- **********
         DO 260 II = M1, K
            I = M1 + K - II
            IF (XU .GE. RV4(I)) GO TO 260
            XU = RV4(I)
            GO TO 280
  260    CONTINUE
C
  280    IF (X0 .GT. RV5(K)) X0 = RV5(K)
C     ********** NEXT BISECTION STEP **********
  300    X1 = (XU + X0) * 0.5
         IF ((X0 - XU) .LE. (2.0 * MACHEP *
     X       (ABS(XU) + ABS(X0)) + ABS(EPS1))) GO TO 420
C     ********** IN-LINE PROCEDURE FOR STURM SEQUENCE **********
  320    S = P - 1
         U = 1.0
C
         DO 340 I = P, Q
            IF (U .NE. 0.0) GO TO 325
            V = ABS(E(I)) / MACHEP
            GO TO 330
  325       V = E2(I) / U
  330       U = D(I) - X1 - V
            IF (U .LT. 0.0) S = S + 1
  340    CONTINUE
C
         GO TO (60,80,200,220,360), ISTURM
C     ********** REFINE INTERVALS **********
  360    IF (S .GE. K) GO TO 400
         XU = X1
         IF (S .GE. M1) GO TO 380
         RV4(M1) = X1
         GO TO 300
  380    RV4(S+1) = X1
         IF (RV5(S) .GT. X1) RV5(S) = X1
         GO TO 300
  400    X0 = X1
         GO TO 300
```

```
C     ********** K-TH EIGENVALUE FOUND **********
  420    RV5(K) = X1
      K = K - 1
      IF (K .GE. M1) GO TO 250
C     ********** ORDER EIGENVALUES TAGGED WITH THEIR
C                SUBMATRIX ASSOCIATIONS **********
  900 S = R
      R = R + M2 - M1 + 1
      J = 1
      K = M1
C
      DO 920 L = 1, R
         IF (J .GT. S) GO TO 910
         IF (K .GT. M2) GO TO 940
         IF (RV5(K) .GE. W(L)) GO TO 915
C
         DO 905 II = J, S
            I = L + S - II
            W(I+1) = W(I)
            IND(I+1) = IND(I)
  905    CONTINUE
C
  910    W(L) = RV5(K)
         IND(L) = TAG
         K = K + 1
         GO TO 920
  915    J = J + 1
  920 CONTINUE
C
  940 IF (Q .LT. N) GO TO 100
      GO TO 1001
C     ********** SET ERROR -- INTERVAL CANNOT BE FOUND CONTAINING
C                EXACTLY THE DESIRED EIGENVALUES **********
  980 IERR = 3 * N + ISTURM
 1001 LB = T1
      UB = T2
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F293-2 TSTURM

A Fortran IV Subroutine to Determine Some Eigenvalues and Eigenvectors of a Symmetric Tridiagonal Matrix.

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV subroutine TSTURM determines those eigenvalues of a symmetric tridiagonal matrix in a specified interval and their corresponding eigenvectors, using Sturm sequencing and inverse iteration.

2. USAGE.

A. Calling Sequence.

The SUBROUTINE statement is

```
SUBROUTINE  TSTURM(NM,N,EPS1,D,E,E2,LB,UB,
                   MM,M,W,Z,IERR,RV1,RV2,RV3,
                   RV4,RV5,RV6)
```

The parameters are discussed below and the interpretation of working precision for various machines is given in the section discussing certification.

NM — is an integer input variable set equal to the row dimension of the two-dimensional array Z as specified in the DIMENSION statement for Z in the calling program.

N — is an integer input variable set equal to the order of the matrix. N must be not greater than NM.

EPS1 — is a working precision real variable. On input, it specifies an absolute error tolerance for the computed eigenvalues. If MACHEP denotes the relative machine precision, then EPS1 should be chosen so that the accuracy of these eigenvalues is

commensurate with relative perturbations of the order of MACHEP in the matrix elements. If the input EPS1 is non-positive, it is reset to a default value described in section 2C.

D is a working precision real input one-dimensional variable of dimension at least N containing the diagonal elements of the symmetric tridiagonal matrix.

E is a working precision real input one-dimensional variable of dimension at least N containing, in its last N-1 positions, the subdiagonal elements of the symmetric tridiagonal matrix. E(1) is arbitrary.

E2 is a working precision real one-dimensional variable of dimension at least N. On input, the last N-1 positions in this array contain the squares of the subdiagonal elements of the symmetric tridiagonal matrix. E2(1) is arbitrary. On output, E2(1) is set to zero. If any of the elements in E are regarded as negligible, the corresponding elements of E2 are set to zero, and so the matrix splits into a direct sum of submatrices.

LB,UB are working precision real input variables specifying the lower and upper endpoints, respectively, of the interval to be searched for the eigenvalues. If LB is not less than UB, TSTURM computes no eigenvalues or eigenvectors. See section 2C for further details.

MM is an integer input variable set equal to an upper bound for the number of eigenvalues in the interval (LB,UB).

M is an integer output variable set equal to the number of eigenvalues found to lie in the interval (LB,UB).

W is a working precision real output one-dimensional variable of dimension at least MM containing the M eigenvalues of the symmetric tridiagonal matrix in the interval (LB,UB). If the matrix does not split into submatrices, the eigenvalues are in ascending order in W. If the matrix does split, the eigenvalues for each submatrix are in ascending order in W.

Z is a working precision real output two-dimensional variable with row dimension NM and column dimension at least MM. It contains M orthonormal eigenvectors of the symmetric tridiagonal matrix corresponding to the M eigenvalues in W.

IERR is an integer output variable set equal to an error completion code described in section 2B. The normal completion code is zero.

RV1,RV2,RV3
are working precision real temporary one-dimensional variables of dimension at least N used to store the main diagonal and the two adjacent diagonals of the triangular matrix produced in the inverse iteration process.

RV4,RV5,RV6
are working precision real temporary one-dimensional variables of dimension at least N. RV4 and RV5 hold the lower and upper bounds for the eigenvalues in the bisection process. In addition, RV4 holds the multipliers of the Gaussian elimination step in the inverse iteration process. RV6 holds the approximate eigenvectors in this process.

B. Error Conditions and Returns.

If M exceeds MM, TSTURM terminates with no eigenvalues or eigenvectors computed, and IERR is set to 3*N+1. In this case, the output parameter M is the number of eigenvalues found to lie in (LB,UB).

If more than 5 iterations are required to determine an eigenvector, TSTURM terminates with IERR set to 4*N+R, where R is the index of the eigenvector for which the failure occurs. The eigenvalues and eigenvectors in the W and Z arrays should be correct for indices 1,2,...,R-1.

If M does not exceed MM and all the eigenvectors are determined within 5 iterations, IERR is set to zero.

C. Applicability and Restrictions.

To determine some of the eigenvalues and eigenvectors of a full symmetric matrix, TSTURM should be preceded by TRED1 (F277) to provide a suitable symmetric tridiagonal matrix for TSTURM. It should then be followed by TRBAK1 (F279) to back transform the eigenvectors from TSTURM into those of the original matrix.

To determine some of the eigenvalues and eigenvectors of a complex Hermitian matrix, TSTURM should be preceded by HTRIDI (F284) to provide a suitable real symmetric tridiagonal matrix for TSTURM. It should then be followed by HTRIBK (F285) to back transform the eigenvectors from TSTURM into those of the original matrix.

Some of the eigenvalues and eigenvectors of certain non-symmetric tridiagonal matrices can be computed using the combination of FIGI (F280), TSTURM, and BAKVEC (F281). See F280 for the description of this special class of matrices. For these matrices, TSTURM should be preceded by FIGI to provide a suitable symmetric matrix for TSTURM. It should then be followed by BAKVEC to back transform the eigenvectors from TSTURM into those of the original matrix.

The interval (LB,UB) is formally half-open, not including the upper endpoint UB. However, because of rounding errors, the true eigenvalues very close to the endpoints of the interval may be erroneously counted or missed.

The input interval (LB,UB) may be refined internally to a smaller interval known to contain all the eigenvalues in (LB,UB). This insures that TSTURM will not perform unnecessary bisection steps to determine the eigenvalues in (LB,UB).

The computation of the eigenvectors by inverse iteration requires that the precision of the eigenvalues be commensurate with small relative perturbations of the order of MACHEP in the matrix elements. For most symmetric tridiagonal matrices, it is enough that the absolute error EPS1 in the eigenvalues for which eigenvectors are desired be approximately MACHEP times a norm of the matrix. But some matrices require a smaller EPS1, perhaps as small as MACHEP times the eigenvalue of smallest magnitude in the interval (LB,UB). Note, however, that if EPS1 is smaller than necessary, TSTURM will perform unnecessary bisection steps to determine the eigenvalues. For further discussion of EPS1, see reference (2).

If the input EPS1 is non-positive, TSTURM resets it, for each submatrix, to -MACHEP times the 1-norm of the submatrix and uses its magnitude. This value is considered adequate for most symmetric tridiagonal matrices and is recommended for use generally. If TSTURM terminates with a vector error exit, a smaller EPS1 may be more successful.

3. DISCUSSION OF METHOD AND ALGORITHM.

The eigenvalues are determined by the method of bisection applied to the Sturm sequence and the eigenvectors are determined by inverse iteration.

The calculations proceed as follows. First, the subdiagonal elements are tested for negligibility. If an element is considered negligible, its square is set to zero, and so the matrix splits into a direct sum of submatrices. Then, the Sturm sequence for the entire matrix is evaluated at UB and LB giving the number of eigenvalues of the matrix less than UB and LB respectively. The difference is the number of eigenvalues in (LB,UB).

Next, a submatrix is examined for its eigenvalues in the interval (LB,UB). The Gerschgorin interval, known to contain all the eigenvalues, is determined and used to refine the input interval (LB,UB). If the input EPS1 is non-positive, it is reset to the default value described in section 2C. Then, subintervals, each enclosing an eigenvalue in (LB,UB), are shrunk using a bisection process until the endpoints of each subinterval are close enough to be accepted as an eigenvalue of the submatrix. Here the endpoints of each subinterval are close enough when they differ by less than MACHEP times twice the sum of the magnitudes of the endpoints plus the absolute error tolerance EPS1.

The eigenvectors of the same submatrix are then computed by inverse iteration. First, the LU decomposition of the submatrix with an approximate eigenvalue subtracted from its diagonal elements is achieved by Gaussian elimination using partial pivoting. The multipliers defining the lower triangular matrix L are stored in the temporary array RV4 and the upper triangular matrix U is stored in the three temporary arrays RV1, RV2, and RV3. Saving these quantities in RV1, RV2, RV3, and RV4 avoids repeating the LU decomposition if further iterations are required. An approximate vector, stored in RV6, is computed starting from an initial vector, and the norm of the approximate vector is compared with a norm of the submatrix to determine whether the growth is sufficient to accept it as an eigenvector. If this vector is accepted, its Euclidean norm

is made 1. If the growth is not sufficient, this vector is used as the initial vector in computing the next approximate vector. This iteration process is repeated at most 5 times.

Eigenvectors computed in the above way corresponding to well-separated eigenvalues of this submatrix will be orthogonal. However, eigenvectors corresponding to close eigenvalues of this submatrix may not be satisfactorily orthogonal. Hence, to insure orthogonal eigenvectors, each approximate vector is made orthogonal to those previously computed eigenvectors whose eigenvalues are close to the current eigenvalue. If the orthogonalization process produces a zero vector, a column of the identity matrix is used as an initial vector for the next iteration.

Identical eigenvalues are perturbed slightly in an attempt to obtain independent eigenvectors. These perturbations are not recorded in the output eigenvalue array W.

The above steps are repeated on each submatrix until all the eigenvalues in the interval (LB,UB) and their corresponding eigenvectors are computed.

This subroutine is a translation of the Algol procedure TRISTURM written and discussed in detail by Peters and Wilkinson (1).

4. REFERENCES.

1) Peters, G. and Wilkinson, J.H., The Calculation of Specified Eigenvectors by Inverse Iteration, Handbook for Automatic Computation, Volume II, Linear Algebra, J. H. Wilkinson - C. Reinsch, Contribution II/18, 418-439, Springer-Verlag, 1971.

2) Barth, W., Martin, R.S., and Wilkinson, J.H., Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection, Num. Math. 9,386-393 (1967).

5. CHECKOUT.

A. Test Cases.

See the section discussing testing of the codes for complex Hermitian, real symmetric, real symmetric tridiagonal, and certain real non-symmetric tridiagonal matrices.

B. Accuracy.

The subroutine TSTURM is numerically stable (1,2); that is, the computed eigenvalues are close to those of the original matrix. In addition, they are the exact eigenvalues of a matrix close to the original real symmetric tridiagonal matrix and the computed eigenvectors are close (but not necessarily equal) to the eigenvectors of that matrix.

```
      SUBROUTINE TSTURM(NM,N,EPS1,D,E,E2,LB,UB,MM,M,W,Z,
     X                  IERR,RV1,RV2,RV3,RV4,RV5,RV6)
C
      INTEGER I,J,K,M,N,P,Q,R,S,II,IP,JJ,MM,M1,M2,NM,ITS,
     X        IERR,GROUP,ISTURM
      REAL D(N),E(N),E2(N),W(MM),Z(NM,MM),
     X       RV1(N),RV2(N),RV3(N),RV4(N),RV5(N),RV6(N)
      REAL U,V,LB,T1,T2,UB,UK,XU,X0,X1,EPS1,EPS2,EPS3,EPS4,
     X       NORM,MACHEP
      REAL SQRT,ABS,AMAX1,AMIN1,FLOAT
C
C     ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C                THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C
C                **********
      MACHEP = ?
C
      IERR = 0
      T1 = LB
      T2 = UB
C     ********** LOOK FOR SMALL SUB-DIAGONAL ENTRIES **********
      DO 40 I = 1, N
         IF (I .EQ. 1) GO TO 20
         IF (ABS(E(I)) .GT. MACHEP * (ABS(D(I)) + ABS(D(I-1))))
     X      GO TO 40
   20    E2(I) = 0.0
   40 CONTINUE
C     ********** DETERMINE THE NUMBER OF EIGENVALUES
C                IN THE INTERVAL **********
      P = 1
      Q = N
      X1 = UB
      ISTURM = 1
      GO TO 320
   60 M = S
      X1 = LB
      ISTURM = 2
      GO TO 320
   80 M = M - S
      IF (M .GT. MM) GO TO 980
      Q = 0
      R = 0
C     ********** ESTABLISH AND PROCESS NEXT SUBMATRIX, REFINING
C                INTERVAL BY THE GERSCHGORIN BOUNDS **********
  100 IF (R .EQ. M) GO TO 1001
      P = Q + 1
      XU = D(P)
      X0 = D(P)
      U = 0.0
```

```
C
      DO 120 Q = P, N
         X1 = U
         U = 0.0
         V = 0.0
         IF (Q .EQ. N) GO TO 110
         U = ABS(E(Q+1))
         V = E2(Q+1)
  110    XU = AMIN1(D(Q)-(X1+U),XU)
         X0 = AMAX1(D(Q)+(X1+U),X0)
         IF (V .EQ. 0.0) GO TO 140
  120 CONTINUE
C
  140 X1 = AMAX1(ABS(XU),ABS(X0)) * MACHEP
      IF (EPS1 .LE. 0.0) EPS1 = -X1
      IF (P .NE. Q) GO TO 180
C     ********** CHECK FOR ISOLATED ROOT WITHIN INTERVAL **********
      IF (T1 .GT. D(P) .OR. D(P) .GE. T2) GO TO 940
      R = R + 1
C
      DO 160 I = 1, N
  160 Z(I,R) = 0.0
C
      W(R) = D(P)
      Z(P,R) = 1.0
      GO TO 940
  180 X1 = X1 * FLOAT(Q-P+1)
      LB = AMAX1(T1,XU-X1)
      UB = AMIN1(T2,X0+X1)
      X1 = LB
      ISTURM = 3
      GO TO 320
  200 M1 = S + 1
      X1 = UB
      ISTURM = 4
      GO TO 320
  220 M2 = S
      IF (M1 .GT. M2) GO TO 940
C     ********** FIND ROOTS BY BISECTION **********
      X0 = UB
      ISTURM = 5
C
      DO 240 I = M1, M2
         RV5(I) = UB
         RV4(I) = LB
  240 CONTINUE
C     ********** LOOP FOR K-TH EIGENVALUE
C                FOR K=M2 STEP -1 UNTIL M1 DO --
C                (-DO- NOT USED TO LEGALIZE COMPUTEE-GO-TO) **********
      K = M2
  250    XU = LB
```

```
C     ********** FOR I=K STEP -1 UNTIL M1 DO -- **********
         DO 260 II = M1, K
            I = M1 + K - II
            IF (XU .GE. RV4(I)) GO TO 260
            XU = RV4(I)
            GO TO 280
  260    CONTINUE
C
  280    IF (X0 .GT. RV5(K)) X0 = RV5(K)
C     ********** NEXT BISECTION STEP **********
  300    X1 = (XU + X0) * 0.5
         IF ((X0 - XU) .LE. (2.0 * MACHEP *
     X      (ABS(XU) + ABS(X0)) + ABS(EPS1))) GO TO 420
C     ********** IN-LINE PROCEDURE FOR STURM SEQUENCE **********
  320    S = P - 1
         U = 1.0
C
         DO 340 I = P, Q
            IF (U .NE. 0.0) GO TO 325
            V = ABS(E(I)) / MACHEP
            GO TO 330
  325       V = E2(I) / U
  330       U = D(I) - X1 - V
            IF (U .LT. 0.0) S = S + 1
  340    CONTINUE
C
         GO TO (60,80,200,220,360), ISTURM
C     ********** REFINE INTERVALS **********
  360    IF (S .GE. K) GO TO 400
         XU = X1
         IF (S .GE. M1) GO TO 380
         RV4(M1) = X1
         GO TO 300
  380    RV4(S+1) = X1
         IF (RV5(S) .GT. X1) RV5(S) = X1
         GO TO 300
  400    X0 = X1
         GO TO 300
C     ********** K-TH EIGENVALUE FOUND **********
  420    RV5(K) = X1
      K = K - 1
      IF (K .GE. M1) GO TO 250
C     ********** FIND VECTORS BY INVERSE ITERATION **********
      NORM = ABS(D(P))
      IP = P + 1
C
      DO 500 I = IP, Q
  500 NORM = NORM + ABS(D(I)) + ABS(E(I))
```

```
C      ********** EPS2 IS THE CRITERION FOR GROUPING,
C                 EPS3 REPLACES ZERO PIVOTS AND EQUAL
C                 ROOTS ARE MODIFIED BY EPS3,
C                 EPS4 IS TAKEN VERY SMALL TO AVOID OVERFLOW **********
       EPS2 = 1.0E-3 * NORM
       EPS3 = MACHEP * NORM
       UK = FLOAT(Q-P+1)
       EPS4 = UK * EPS3
       UK = EPS4 / SQRT(UK)
       GROUP = 0
       S = P
C
       DO 920 K = M1, M2
          R = R + 1
          ITS = 1
          W(R) = RV5(K)
          X1 = RV5(K)
C      ********** LOOK FOR CLOSE OR COINCIDENT ROOTS **********
          IF (K .EQ. M1) GO TO 520
          IF (X1 - X0 .GE. EPS2) GROUP = -1
          GROUP = GROUP + 1
          IF (X1 .LE. X0) X1 = X0 + EPS3
C      ********** ELIMINATION WITH INTERCHANGES AND
C                 INITIALIZATION OF VECTOR **********
  520     V = 0.0
C
          DO 580 I = P, Q
             RV6(I) = UK
             IF (I .EQ. P) GO TO 560
             IF (ABS(E(I)) .LT. ABS(U)) GO TO 540
             XU = U / E(I)
             RV4(I) = XU
             RV1(I-1) = E(I)
             RV2(I-1) = D(I) - X1
             RV3(I-1) = 0.0
             IF (I .NE. Q) RV3(I-1) = E(I+1)
             U = V - XU * RV2(I-1)
             V = -XU * RV3(I-1)
             GO TO 580
  540        XU = E(I) / U
             RV4(I) = XU
             RV1(I-1) = U
             RV2(I-1) = V
             RV3(I-1) = 0.0
  560        U = D(I) - X1 - XU * V
             IF (I .NE. Q) V = E(I+1)
  580     CONTINUE
C
          IF (U .EQ. 0.0) U = EPS3
          RV1(Q) = U
          RV2(Q) = 0.0
          RV3(Q) = 0.0
```

```
C     ********** BACK SUBSTITUTION
C                FOR I=Q STEP -1 UNTIL P DO -- **********
  600    DO 620 II = P, Q
            I = P + Q - II
            RV6(I) = (RV6(I) - U * RV2(I) - V * RV3(I)) / RV1(I)
            V = U
            U = RV6(I)
  620    CONTINUE
C     ********** ORTHOGONALIZE WITH RESPECT TO PREVIOUS
C                MEMBERS OF GROUP **********
         IF (GROUP .EQ. 0) GO TO 700
C
         DO 680 JJ = 1, GROUP
            J = R - GROUP - 1 + JJ
            XU = 0.0
C
            DO 640 I = P, Q
  640       XU = XU + RV6(I) * Z(I,J)
C
            DO 660 I = P, Q
  660       RV6(I) = RV6(I) - XU * Z(I,J)
C
  680    CONTINUE
C
  700    NORM = 0.0
C
         DO 720 I = P, Q
  720    NORM = NORM + ABS(RV6(I))
C
         IF (NORM .GE. 1.0) GO TO 840
C     ********** FORWARD SUBSTITUTION **********
         IF (ITS .EQ. 5) GO TO 960
         IF (NORM .NE. 0.0) GO TO 740
         RV6(S) = EPS4
         S = S + 1
         IF (S .GT. Q) S = P
         GO TO 780
  740    XU = EPS4 / NORM
C
         DO 760 I = P, Q
  760    RV6(I) = RV6(I) * XU
C     ********** ELIMINATION OPERATIONS ON NEXT VECTOR
C                ITERATE **********
  780    DO 820 I = IP, Q
            U = RV6(I)
C     ********** IF RV1(I-1) .EQ. E(I), A ROW INTERCHANGE
C                WAS PERFORMED EARLIER IN THE
C                TRIANGULARIZATION PROCESS **********
            IF (RV1(I-1) .NE. E(I)) GO TO 800
            U = RV6(I-1)
            RV6(I-1) = RV6(I)
  800       RV6(I) = U - RV4(I) * RV6(I-1)
  820    CONTINUE
```

```
C
         ITS = ITS + 1
         GO TO 600
C     ********** NORMALIZE SO THAT SUM OF SQUARES IS
C                1 AND EXPAND TO FULL ORDER **********
  840    U = 0.0
C
         DO 860 I = P, Q
  860    U = U + RV6(I)**2
C
         XU = 1.0 / SQRT(U)
C
         DO 880 I = 1, N
  880    Z(I,R) = 0.0
C
         DO 900 I = P, Q
  900    Z(I,R) = RV6(I) * XU
C
         X0 = X1
  920 CONTINUE
C
  940 IF (Q .LT. N) GO TO 100
      GO TO 1001
C     ********** SET ERROR -- NON-CONVERGED EIGENVECTOR **********
  960 IERR = 4 * N + R
      GO TO 1001
C     ********** SET ERROR -- UNDERESTIMATE OF NUMBER OF
C                EIGENVALUES IN INTERVAL **********
  980 IERR = 3 * N + 1
 1001 LB = T1
      UB = T2
      RETURN
      END
```

NATS PROJECT

EIGENSYSTEM SUBROUTINE PACKAGE (EISPACK)

F299-2 EISPAC

A Control Program for the Eigensystem Package
(F269 to F298 and F220 to F247).

May, 1972
July, 1975

1. PURPOSE.

The Fortran IV and OS/360-370 assembly language subroutine EISPAC (together with its associated "keyword" entry points) is designed to simplify the solution of the standard and generalized matrix eigenproblems using the subroutines in the eigensystem package (F269-F298, F220-F247). It thus may be used to compute some or all of the eigenvalues, with or without eigenvectors, of complex general, complex Hermitian, real general, real symmetric, real symmetric tridiagonal, certain real non-symmetric tridiagonal, and real symmetric band matrices; and it may also be used to compute some or all of the eigenvalues, with or without eigenvectors, for the real symmetric generalized eigenproblem, or all of the eigenvalues, with or without eigenvectors, for the real non-symmetric generalized eigenproblem. EISPAC offers the following advantages if you wish to solve an eigenproblem:

1. You describe the problem to EISPAC in simple, familiar terms.
2. Using your description, EISPAC automatically selects subroutines to solve your problem and executes them in the proper order and with parameters passed from one to another in the proper way. In general, EISPAC selects the sequence of subroutines in the eigensystem package which will solve your problem as rapidly as possible with reasonable assurance of stability in the calculations.
3. EISPAC loads each selected subroutine only as it is required, thus making available for data storage as much memory as possible. On the other hand, if sufficient memory is available, the subroutines used are retained in memory and are not reloaded when you repeatedly call EISPAC to solve the same problem. (See section 3 below.)
4. EISPAC allocates and frees any necessary auxiliary storage automatically.

5. Use of EISPAC minimizes the number of changes you must make to your program if you wish to solve a slight variation of your original problem or if you wish to take advantage of new methods when they are introduced.
6. EISPAC is called as a subroutine from your own driver program. Thus the matrix whose eigensystem is being computed may itself be the result of other computations in your program, and the computed eigensystem may conveniently be used in further calculations.

Thus EISPAC relieves you of having to develop a detailed knowledge of the applicability, efficiency, and calling sequences of the individual EISPACK subroutines.

2. USAGE.

A. Calling Sequence.

EISPAC employs variable length parameter lists and "keyword" parameters to simplify specification of your problem and to group related subparameters. An overview of the calls to EISPAC is given by the prototype calls below; specific calls for different classes of matrices are discussed in section 2.B.

The prototype call to EISPAC for the standard eigenproblem is:

```
CALL EISPAC (NM, N, MATRIX (...), BAND (...), VALUES
   (...), VECTOR (...), METHOD (...), ERROR (...),
   SUBR (...))
```

where

NM — is an integer variable which supplies the row (first) dimension of any two-dimensional arrays appearing in the call to EISPAC (as that dimension is specified in the DIMENSION statements for them in the calling program). For example, if your program contains arrays dimensioned: A(50,50), Z(50,50) which appear in the call to EISPAC, and the order of the eigenproblem being solved is 25, then NM = 50 and N = 25.

N is an integer variable which supplies the order of the input matrix. N is checked to verify that it is neither greater than NM nor less than 1.

MATRIX, BAND, VALUES, VECTOR, METHOD, ERROR, SUBR are keyword parameters. They accept sets of subparameters which define the eigenproblem being solved. (These subparameters are discussed in detail in section 2.C below.) Keyword parameters not needed to specify a particular problem are omitted from the call to EISPAC; those appearing may appear in any order.

The prototype call to EISPAC for the generalized eigenproblem is:

```
CALL EISPAC (NM, N, MATA (...), MATB (...), VALUES
   (...), VECTOR (...), ERROR (...), SUBR (...))
```

where MATA and MATB replace MATRIX as keyword parameters. Note: MATRIX and MATA are alternate forms of the same keyword and can be used interchangeably.

To use EISPAC, you must include a private library DD card for EISPACLB (the dynamic load library, see section 3 below) in the GO-step of your job. This card associates EISPACLB with the dataset where the load modules for EISPAC and the eigensystem package subroutines reside. If you are using a typical Fortran cataloged procedure, this card is:

```
//GO.EISPACLB DD DISP=SHR,DSNAME=...
```

Similarly, if you are using the TSO time sharing system, you should execute the command:

```
ALLOC FILE(EISPACLB) DATASET(...) SHR
```

before executing your program.

Finally, if you wish to use EISPAC from a PL/I (Checkout or Optimizer) program by means of the interlanguage communication facility, you should write a simple Fortran subroutine which is called from your PL/I program and which in turn calls EISPAC as described below. This is necessary because EISPAC makes certain (non-standard) assumptions about the Fortran environment of the program calling it which may not be satisfied

when the program is written in PL/I. The Fortran program calling EISPAC should have as parameters the variables to be passed to EISPAC; you should take care that the vectors and arrays appear in DIMENSION statements in it. Thus for the example concluding the section on real symmetric matrices in section 2.B below, the Fortran subroutine called by your PL/I program should look like:

```
SUBROUTINE EISCAL (N, A, W, Z)
INTEGER N
REAL*8 A(N,N), W(N), Z(N,N)
CALL EISPAC (N, N, MATRIX ('REAL, A, 'SYMMETRIC'),
   VALUES (W), VECTOR (Z))
RETURN
END
```

This example assumes that the PL/I arrays corresponding to the matrices A and Z are dimensioned exactly NxN, and hence that the parameter NM can be replaced by N. In the PL/I calling program, EISCAL would be declared:

```
DCL EISCAL ENTRY (FIXED BINARY(31,0),
                  (*,*) FLOAT DECIMAL(16),
                  (*)   FLOAT DECIMAL(16),
                  (*,*) FLOAT DECIMAL(16))
                  OPTIONS (FORTRAN) EXTERNAL;
```

You must provide a DD card or allocation for FT06F001, the standard Fortran output file, because Fortran automatically tries to open it. This file should be associated with the same dataset as the output from the PL/I program, so that EISPAC error messages, if any, appear with it. Finally, if you are using the PL/I Checkout compiler, you need to specify the option SIZE(-nK) to the execution step to reserve dynamic storage for EISPAC; at a minimum n=14, and it must be large enough to accommodate any temporary arrays that are allocated. (An insufficiently large value of n will result in either of the system completion codes 804 or 80A (see section 2.D).)

The use of EISPAC to solve various kinds of eigenproblems is illustrated in the following section. You may wish to read first the examples for the type of matrix you have and then consult section 2.C for further details about the keyword subparameters. (Note that in these examples the continuation symbol which must appear in column 6 of a Fortran continuation card has been omitted.)

B. EISPAC Examples for Different Classes of Matrices.

Complex General Matrix

The basic call, which finds all of the eigenvalues (WR, WI) of the complex general matrix (AR, AI), is:

```
CALL EISPAC (NM, N, MATRIX ('COMPLEX', AR, AI),
   VALUES (WR, WI))
```

If you wish to find all of the eigenvectors (ZR, ZI), add: VECTOR (ZR, ZI).

If you wish to find all of the eigenvalues and only selected eigenvectors, add: VECTOR (ZR, ZI, MM, M, SELECT).

If you prefer not to balance the matrix, add: METHOD ('NO', 'BALANCE').

If you wish to use elementary instead of unitary similarities, add: METHOD ('ELEMENTARY').

Thus a call to find all of the eigenvalues and some of the eigenvectors of a complex general matrix is:

```
CALL EISPAC (NM, N, MATRIX ('COMPLEX', AR, AI),
   VALUES (WR, WI), VECTOR (ZR, ZI, MM, M,
   SELECT))
```

Complex Hermitian Matrix

Recall that a complex Hermitian matrix is a matrix which is equal to its complex conjugate transpose. If you wish to solve an eigenproblem for a complex symmetric matrix (a matrix which is equal to its transpose without conjugation), you must follow the complex general matrix examples.

The basic call, which finds all of the eigenvalues W of the complex Hermitian matrix (AR, AI), is:

```
CALL EISPAC (NM, N, MATRIX ('COMPLEX', AR, AI,
   'HERMITIAN'), VALUES (W))
```

If the complex Hermitian matrix is packed into a single two-dimensional array, HP, substitute for MATRIX ('COMPLEX', AR, AI, 'HERMITIAN'): MATRIX ('COMPLEX', HP, HP, 'HERMITIAN', 'PACKED').

If you wish to find only the eigenvalues in the interval (RLB,RUB), substitute for VALUES (W) (see note 3): VALUES (W, MM, M, RLB, RUB).

Alternatively, if you wish to find exactly M eigenvalues, starting from the M11-th smallest one, substitute for VALUES (W): VALUES (W, M11, M). (See notes 1 and 3.)

If you wish to find the eigenvectors (ZR, ZI) corresponding to the eigenvalues found, add: VECTOR (ZR, ZI).

Thus a call to find a few eigenvalues and their corresponding eigenvectors of a complex Hermitian matrix is:

```
CALL EISPAC (NM, N, MATRIX ('COMPLEX', AR, AI,
   'HERMITIAN'), VALUES (W, MM, M, RLB, RUB),
   VECTOR (ZR, ZI))
```

Real General Matrix

The basic call, which finds all of the eigenvalues (WR, WI) of the real general matrix A, is:

```
CALL EISPAC (NM, N, MATRIX ('REAL', A), VALUES
   (WR, WI))
```

If you wish to find all of the eigenvectors ZP, add: VECTOR (ZP).

If you wish to find all of the eigenvalues and only selected eigenvectors, add: VECTOR (ZP, MM, M, SELECT).

If you prefer not to balance the matrix, add: METHOD ('NO', 'BALANCE').

If you wish to use orthogonal instead of elementary similarities for the reduction to upper Hessenberg form, add (see note 5): METHOD ('ORTHOGONAL').

Thus a call to find all of the eigenvalues of a real general matrix without balancing and using orthogonal similarities is:

```
CALL EISPAC (NM, N, MATRIX ('REAL', A), VALUES
   (WR, WI), METHOD ('NO', 'BALANCE',
   'ORTHOGONAL'))
```

Real Non-Symmetric Tridiagonal Matrix

The following examples apply only if the products of corresponding off-diagonal elements of the tridiagonal matrix all are non-negative. (Read note 4 below for an additional restriction if you wish to compute eigenvectors.) Otherwise the real general matrix examples must be followed.

The basic call, which finds all of the eigenvalues W of the real tridiagonal matrix UST, is:

```
CALL EISPAC (NM, N, MATRIX ('REAL', UST,
   'TRIDIAGONAL'), VALUES (W))
```

If you wish to find only the eigenvalues in the interval (RLB,RUB), substitute for VALUES (W) (see note 3): VALUES (W, MM, M, RLB, RUB).

Alternatively, if you wish to find exactly M eigenvalues, starting from the M11-th smallest one, substitute for VALUES (W): VALUES (W, M11, M). (See notes 1 and 3.)

If you wish to find the eigenvectors Z corresponding to the eigenvalues found, add: VECTOR (Z).

Thus a call to find exactly M eigenvalues, starting from the M11-th smallest one, of a real non-symmetric tridiagonal matrix is:

```
CALL EISPAC (NM, N, MATRIX ('REAL', UST,
   'TRIDIAGONAL'), VALUES (W, M11, M))
```

Real Symmetric Matrix

The basic call, which finds all of the eigenvalues W of the real symmetric matrix A, is (see note 2):

```
CALL EISPAC (NM, N, MATRIX ('REAL', A,
   'SYMMETRIC'), VALUES (W))
```

If the real symmetric matrix is packed into a single one-dimensional array, SP, substitute for MATRIX ('REAL', A, 'SYMMETRIC'): MATRIX ('REAL', SP, 'SYMMETRIC', 'PACKED').

If you wish to find only the eigenvalues in the interval (RLB,RUB), substitute for VALUES (W) (see note 3): VALUES (W, MM, M, RLB, RUB).

Alternatively, if you wish to find exactly M eigenvalues, starting from the M11-th smallest one, substitute for VALUES (W): VALUES (W, M11, M). (See notes 1 and 3.)

If you wish to find the eigenvectors Z corresponding to the eigenvalues found, add: VECTOR (Z).

Thus a call to find all of the eigenvalues and eigenvectors of a real symmetric matrix is:

```
CALL EISPAC (NM, N, MATRIX ('REAL', A,
   'SYMMETRIC'), VALUES (W), VECTOR (Z))
```

Real Symmetric Tridiagonal Matrix

The basic call, which finds all of the eigenvalues W of the real symmetric tridiagonal matrix ST, is (see note 2):

```
CALL EISPAC (NM, N, MATRIX ('REAL', ST,
   'SYMMETRIC', 'TRIDIAGONAL'), VALUES (W))
```

If you wish to find only the eigenvalues in the interval (RLB,RUB), substitute for VALUES (W) (see note 3): VALUES (W, MM, M, RLB, RUB).

Alternatively, if you wish to find exactly M eigenvalues, starting from the M11-th smallest one, substitute for VALUES (W): VALUES (W, M11, M). (See notes 1 and 3.)

If you wish to find the eigenvectors Z corresponding to the eigenvalues found, add: VECTOR (Z).

Thus a call to find some of the eigenvalues of a real symmetric tridiagonal matrix is:

```
CALL EISPAC (NM, N, MATRIX ('REAL', ST,
   'SYMMETRIC', 'TRIDIAGONAL'), VALUES (W, MM,
   M, RLB, RUB))
```

Real Symmetric Band Matrix

The basic call, which finds all of the eigenvalues W of the real symmetric band matrix SB, is (see notes 2 and 6):

```
CALL EISPAC (NM, N, MATRIX ('REAL', SB,
   'SYMMETRIC'), BAND (MB), VALUES (W))
```

If you wish to find only the eigenvalues in the interval (RLB,RUB), substitute for VALUES (W) (see note 3): VALUES (W, MM, M, RLB, RUB).

Alternatively, if you wish to find exactly M eigenvalues, starting from the M11-th smallest one, substitute for VALUES (W): VALUES (W, M11, M). (See notes 1 and 3.)

If you wish to find the eigenvectors Z corresponding to the eigenvalues found, add: VECTOR (Z).

Thus a call to find some of the eigenvalues of a real symmetric band matrix is:

```
CALL EISPAC (NM, N, MATRIX ('REAL', SB,
   'SYMMETRIC'), BAND (MB), VALUES (W, MM, M,
   RLB, RUB))
```

Generalized Real Symmetric Matrix Systems

There are three forms of the real symmetric generalized eigenproblem that can be solved with EISPAC: namely, the forms Ax = (lambda)Bx, ABx = (lambda)x, and BAx = (lambda)x, where A and B are both symmetric and B is positive definite.

The basic call, which finds all of the eigenvalues W of the real symmetric generalized system Ax = (lambda)Bx, is (see notes 2 and 7):

```
CALL EISPAC (NM, N, MATA ('REAL', A,
   'SYMMETRIC'), MATB ('REAL', B, 'SYMMETRIC',
   'POSITIVE DEFINITE'), VALUES (W))
```

If you wish to find only the eigenvalues in the interval (RLB,RUB), substitute for VALUES (W) (see note 3): VALUES (W, MM, M, RLB, RUB).

Alternatively, if you wish to find exactly M eigenvalues, starting from the M11-th smallest one, substitute for VALUES (W): VALUES (W, M11, M). (See notes 1 and 3.)

If you wish to find the eigenvectors Z corresponding to the eigenvalues found, add: VECTOR (Z).

If the problem is of the form ABx = (lambda)x or BAx = (lambda)x, supply 'ABX=LX' or 'BAX=LX', respectively, to MATB (in a position after the B parameter).

Thus a call to find exactly M eigenvalues, starting from the M11-th smallest one, and the corresponding eigenvectors of the real symmetric generalized system ABx = (lambda)x, is:

```
CALL EISPAC (NM, N, MATA ('REAL', A,
   'SYMMETRIC'), MATB ('REAL', B, 'SYMMETRIC',
   'POSITIVE DEFINITE', 'ABX=LX'), VALUES (W,
   M11, M), VECTOR (Z))
```

Generalized Real Non-Symmetric Matrix System

The basic call, which finds all of the eigenvalues (ALPHAR/BETA, ALPHAI/BETA) for the real non-symmetric generalized eigenproblem Ax = (lambda)Bx, is (see note 8):

```
CALL EISPAC (NM, N, MATA ('REAL', A), MATB
   ('REAL', B), VALUES (ALPHAR, ALPHAI, BETA))
```

If you wish to find all of the eigenvectors, ZP, of the system, add: VECTOR (ZP).

Thus a call to find all of the eigenvalues and eigenvectors of a real non-symmetric generalized system is:

```
CALL EISPAC (NM, N, MATA ('REAL', A), MATB
   ('REAL', B), VALUES (ALPHAR, ALPHAI, BETA),
   VECTOR (ZP))
```

C. Meanings of Keyword Subparameters.

Subparameters for MATRIX, MATA, and MATB:

'REAL' specifies that the matrix whose eigensystem is to be computed is real. (See note 9.)

'COMPLEX' specifies that the matrix whose eigensystem is to be computed is complex. (See note 9.)

AR, AI are long precision real two-dimensional variables with row dimension NM and column dimension at least N. They supply, respectively, the real and imaginary parts of the complex matrix whose eigensystem is to be computed. In the non-Hermitian case, the information in the full AR and AI arrays specifies the matrix, and all of it is destroyed. In the Hermitian case, the information in the full lower triangle of AR and the strict lower triangle of AI specifies the matrix, and is destroyed as well as the diagonal of AI; the full upper triangle of AR and the strict upper triangle of AI are left unaltered.

A, B are long precision real two-dimensional variables with row dimension NM and column dimension at least N. They supply the real matrices whose eigensystem is to be computed. (For the standard eigenproblem, only A is supplied.) In the non-symmetric case, the information in the full A and B arrays specifies the matrices, and all of it is destroyed. In the symmetric standard case, the information in the full lower triangle of A specifies the matrix; that in the strict lower triangle may be destroyed, and that in the full upper triangle is left

unaltered. In the symmetric generalized case, the information in the full upper triangles of A and B specifies the matrices; the full lower triangle of A and the strict lower triangle of B are destroyed, and the strict upper triangle of A and the full upper triangle of B are left unaltered.

UST is a long precision real two-dimensional variable with row dimension NM and column dimension at least 3. It supplies a real non-symmetric tridiagonal matrix whose subdiagonal elements are stored in the last N-1 positions of the first column, whose diagonal elements are stored in the second column, and whose superdiagonal elements are stored in the first N-1 positions of the third column. UST(1,1) and UST(N,3) are arbitrary. None of the information in UST is destroyed. The non-symmetric tridiagonal matrix supplied by UST must have the additional special property that the products of corresponding off-diagonal elements all are non-negative. (Read note 4 for an additional restriction on UST if eigenvectors are being computed.) Violation of these restrictions on UST causes an EISPAC execution phase error (see section 2.D below).

ST is a long precision real two-dimensional variable with row dimension NM and column dimension at least 2. It supplies a real symmetric tridiagonal matrix whose subdiagonal elements are stored in the last N-1 positions of the first column and whose diagonal elements are stored in the second column. ST(1,1) is arbitrary. The information in the subdiagonal (first column) only is destroyed.

HP is a long precision real two-dimensional variable with row dimension NM and column dimension at least N. On input, HP contains the lower triangle of a packed complex Hermitian matrix of order N whose eigensystem is to be computed. If the real part of the matrix is denoted by AR and the imaginary part by AI, then the full

lower triangle of AR should be stored in the full lower triangle of HP, and the strict lower triangle of AI should be stored in the strict upper triangle of HP in transposed form. For example when N=4, HP should contain

```
( AR(1,1)   AI(2,1)   AI(3,1)   AI(4,1) )
( AR(2,1)   AR(2,2)   AI(3,2)   AI(4,2) )
( AR(3,1)   AR(3,2)   AR(3,3)   AI(4,3) )
( AR(4,1)   AR(4,2)   AR(4,3)   AR(4,4) ).
```

All of the information in HP is destroyed.

SP is a long precision real one-dimensional variable with dimension at least N*(N+1)/2. It supplies the lower triangle of a real symmetric matrix A of order N, packed row-wise. For example when N=3, SP should contain

(A(1,1),A(2,1),A(2,2),A(3,1),A(3,2),A(3,3))

where the subscripts of each element refer to the row and column of the element in the standard two-dimensional representation. All of the information in SP is destroyed.

SB is a long precision real two-dimensional variable with row dimension NM and column dimension at least MB. It supplies the lower triangle of a real symmetric matrix A of order N in band form with half bandwidth MB. The lowest subdiagonal of the matrix is stored in the last N+1-MB positions of the first column of SB, the next subdiagonal in the last N+2-MB positions of the second column, further diagonals similarly, and finally the principal diagonal in the N positions of the last column. Contents of storage locations not part of the matrix are arbitrary. For example, when N=5 and MB=3, SB should contain

```
(    *        *      A(1,1) )
(    *      A(2,1)   A(2,2) )
( A(3,1)    A(3,2)   A(3,3) )
( A(4,2)    A(4,3)   A(4,4) )
( A(5,3)    A(5,4)   A(5,5) ).
```

All of the information in SB is destroyed.

'HERMITIAN', 'SYMMETRIC', 'TRIDIAGONAL', 'POSITIVE DEFINITE', 'NEGATIVE DEFINITE', 'PACKED'
specify that the matrix whose eigensystem is to be found has the stated special property. (See also notes 1, 2, and 9.)

'ABX=LX', 'BAX=LX' (as subparameters for MATB)
specify the corresponding variant of the real symmetric generalized eigenproblem.

Subparameter for BAND:

MB is an integer variable which supplies the half bandwidth of a real symmetric band matrix. It is defined as the number of adjacent diagonals, including the principal diagonal, required to specify the non-zero portion of the lower triangle of the matrix.

Subparameters for VALUES:

WR, WI are long precision real one-dimensional variables of dimension at least N. They receive, respectively, the real and imaginary parts of the N complex eigenvalues computed.

W is a long precision real one-dimensional variable of dimension at least sufficient (N, MM, or M) to hold the eigenvalues. It receives the N (M) eigenvalues computed.

MM is an integer variable which, when it is a subparameter to VALUES, supplies a maximum for the number of eigenvalues expected in the interval (RLB,RUB); more than MM eigenvalues in (RLB,RUB) is an error. The dimension of W, and the column dimension of ZR and ZI or Z if used, must be at least MM.

M is an integer variable which, when it is a subparameter to VALUES, denotes the number of eigenvalues actually stored in W. It is supplied (in conjunction with M11) by the user if a specified number of eigenvalues is to be computed. It is set by the program, if an interval (RLB,RUB) has been specified, to the number of eigenvalues actually found.

RLB, RUB are long precision real variables which define an interval to be searched for eigenvalues. The eigenvalues found lie in the interval (RLB,RUB), which is closed on the left and open on the right; if RLB is not less than RUB, no eigenvalues are found. (RLB,RUB) must not contain more than MM eigenvalues.

M11 is an integer variable which, when it is a subparameter to VALUES, supplies the index of the smallest eigenvalue to be computed.

EPS1 is a long precision real variable which supplies an absolute error tolerance for the computed eigenvalues; if the input value of EPS1 is non-positive, a default value will be used. (See note 3.)

ALPHAR, ALPHAI, BETA
are long precision real one-dimensional variables with dimension at least N which receive the complex eigenvalues for a real non-symmetric generalized eigenproblem. The real part of each eigenvalue is given by ALPHAR(I)/BETA(I) and the imaginary part by ALPHAI(I)/BETA(I). (See note 8.)

'LARGEST', 'SMALLEST'
specify that the M largest or smallest eigenvalues are to be computed if the rational QR method has been selected. In this case substitute for VALUES(W): VALUES (W, M, 'LARGEST') or VALUES (W, M, 'SMALLEST'), respectively. (See notes 1 and 9.)

Subparameters for VECTOR:

ZR, ZI are long precision real two-dimensional variables with row dimension NM and column dimension at least sufficient (N, MM, or M) to hold the eigenvectors. They receive, respectively, the real and imaginary parts of the N (M) eigenvectors computed.

ZP is a long precision real two-dimensional variable with row and column dimensions as described for ZR and ZI. It receives N (M) columns which represent eigenvectors of a real general matrix or matrix system. To conserve storage, these eigenvectors are stored in ZP in packed form:

corresponding to a real eigenvalue, one column representing the real eigenvector is stored; corresponding to a complex conjugate pair of eigenvalues, two consecutive columns representing the real and imaginary parts of the eigenvector corresponding to the first, or first flagged (see SELECT below), of the pair are stored. For example, suppose all of the eigenvectors of a real general matrix are being computed. If the K-th eigenvalue is real (WI(K) = 0) then ZP(K) is the corresponding eigenvector. If the K-th and (K+1)-th eigenvalues are a complex conjugate pair (with WI(K) .GT. 0) then ZP(K) is the real part and ZP(K+1) is the imaginary part of the eigenvector corresponding to the K-th eigenvalue. The conjugate of this vector is the eigenvector for the conjugate eigenvalue. (Example Fortran statements for unpacking the eigenvectors may be found in section 2.3.2 of (6).)

Z — is a long precision real two-dimensional variable with row and column dimensions as described for ZR and ZI. It receives the N (M) eigenvectors computed. When all of the eigenvectors of a real symmetric (full) matrix or generalized matrix system are being computed, the array A which supplies the input matrix may be used for Z; the eigenvectors then overwrite the input matrix.

MM — is an integer variable which, when it is a subparameter to VECTOR, supplies the maximum number of columns of ZR and ZI or ZP which will be used. The column dimension of ZR and ZI or ZP must be at least MM.

M — is an integer variable which, when it is a subparameter to VECTOR, receives the number of columns used to store eigenvectors in ZR and ZI or ZP.

SELECT — is a logical one-dimensional variable of dimension at least N whose true elements flag those eigenvalues of a real or complex general matrix whose eigenvectors are to be computed. If eigenvectors of a real general matrix are being computed and both of a pair of complex conjugate eigenvalues

are flagged, the second flag is set false and only the eigenvector corresponding to the first of the pair is computed. An error results if MM is smaller than the number of columns required to hold the selected eigenvectors.

Subparameters for METHOD:

(See note 9 concerning alphanumeric parameters.)

'ORTHOGONAL'
specifies that orthogonal similarities (instead of elementary similarities) are to be used to reduce a real general matrix to upper Hessenberg form. Reduction by orthogonal similarities is somewhat slower than that by elementary similarities, but in some cases it may improve the accuracy of the computed eigensystem. In particular, if you wish to calculate the condition numbers for the computation of the eigensystem of a matrix (see (3), p. 199 and (4), pp. 86-89), you should use the condition-number-preserving orthogonal transformations.

'ELEMENTARY'
specifies that elementary similarities (instead of unitary similarities) are to be used on a complex general matrix (the LR rather than the QR transformation). Elementary similarities are somewhat faster than unitary similarities, but in some cases diminish the accuracy of the computed (complex general) eigensystem.

'NO', 'BALANCE'
specifies that a real or complex general matrix is not to be balanced before its eigensystem is computed. In general, balancing improves the accuracy of the computed eigensystem and requires very little additional computation time.

'RATQR' specifies that the rational QR method is to be used to compute a few of the largest or smallest eigenvalues. (See note 1.)

Subparameter for ERROR:

IERROR is an integer variable which receives a value indicating whether certain errors occurred during the solution of the eigensystem. See section 2.D below.

Subparameter for SUBR:

USUB is a user-supplied subroutine with the following parameter list:

```
USUB (SUBRNO, NM, N, AR, AI, WR, WI, ZR,
   ZI, MM, M, RLB, RUB, EPS1, SELECT,
   IDEF, SMLSTV, IERROR, LOW, UPP, BND,
   D, E, E2, IND, INT, ORTR, SCALE, TAU,
   ORTI, M11, NV, MB, BR, DL, ALPHAR,
   ALPHAI, BETA).
```

The keyword parameter SUBR (USUB) may be added to any EISPAC call. (Note that USUB must be mentioned in an EXTERNAL statement in the calling program.) USUB will be called before execution begins and after execution of each numbered routine in the path determined by the parameters to EISPAC and defined in the "EISPAC Eigensystem Path Chart" (1); the number of the routine (zero before execution begins) is passed by the integer variable SUBRNO. USUB may be used to examine intermediate results, to obtain timing information, etc.; its use is quite analogous to that of the procedure "INFORM" described by Rutishauser (5). For the meanings of the parameters it receives, and appropriate dimensions for those requiring them, see (1), (2), and section 2.3.7 of (6).

Note 1: Computing a Few of the Largest or Smallest Eigenvalues.

The recommended procedure for computing a few of the extreme eigenvalues of a real symmetric or complex Hermitian matrix is by appropriate specification of the subparameters M11 and M to VALUES. Alternatively, the rational QR method may be employed, although it has proven less stable numerically. If the rational QR method is selected and you know that the matrix is positive (negative) definite, you may indicate so by adding the subparameter 'POSITIVE DEFINITE' ('NEGATIVE DEFINITE') to MATRIX, e.g.,

```
MATRIX ('REAL', ST, 'SYMMETRIC',
'TRIDIAGONAL', 'POSITIVE DEFINITE'), METHOD
('RATQR'), VALUES (W, M, 'SMALLEST').
```

Computational economy results only when the largest (smallest) eigenvalues of a negative (positive) definite matrix are sought.

Note that subparameter W to VALUES must be of dimension at least N, if the rational QR method is used, even though only M eigenvalues are requested.

Note 2: 'HERMITIAN' for Real Matrices.

For a real matrix, 'HERMITIAN' and 'SYMMETRIC' are synonymous.

Note 3: EPS1 as a Subparameter to VALUES.

EPS1 is a long precision real variable which supplies an absolute error tolerance for the computed eigenvalues when only a few eigenvalues (or a few eigenvalues and their corresponding eigenvectors) are being calculated for the standard or generalized real symmetric problems; or for the non-symmetric generalized problem where all eigenvalues and possibly eigenvectors are being found. If the input value of EPS1 is non-positive, a suitable default value, which is related to the machine precision and the norm of the matrix, is used in place of EPS1; this same default is used if EPS1 is not supplied to EISPAC. The input value, if any, of EPS1 is not altered when the default value is employed; however, for the standard problem only, the (last) default value used may be examined by means of a user subroutine supplied as the subparameter to SUBR (see above). To supply EPS1 to EISPAC, substitute for the recommended call to VALUES: VALUES (W, MM, M, RLB, RUB, EPS1) or VALUES (ALPHAR, ALPHAI, BETA, EPS1) as appropriate.

If eigenvalues only (no eigenvectors) are being computed, EPS1 may be used to control the trade-off between computation time and absolute accuracy of the computed eigenvalues, if less (or more) accuracy than that given by the default value of EPS1 is required.

If eigenvalues and their corresponding eigenvectors are being computed, extreme care must be taken in choosing a value for EPS1 other than the default, since if EPS1 is too large, the eigenvectors may be poor because the eigenvalues are not sufficiently accurate; whereas if EPS1 is smaller than necessary for convergence, the additional accuracy of the computed eigenvalues will not increase the accuracy of the eigenvectors. For further discussion, or in case of difficulty, see section 2.3.3 of (6), and also the NATS subroutine documents for F223 TINVIT, F294 BISECT, F237 TRIDIB, and F239 QZIT (2).

Note 4: Class of Non-Symmetric Tridiagonal Matrices for which Eigenvectors may be Found.

The class of real non-symmetric tridiagonal matrices for which eigenvectors can be computed is somewhat more restricted than that for which eigenvalues can be found. For the eigenvector computation to be successful, not only must the products of corresponding off-diagonal elements all be non-negative, but also when such a product is zero, the corresponding off-diagonal elements must both be zero.

Note 5: 'UNITARY' for Real Matrices.

For a real matrix, 'UNITARY' and 'ORTHOGONAL' are synonymous.

Note 6: Symmetric Band Matrices of Half Bandwidth 2.

If MB is specified as 2 for a real symmetric band matrix, EISPAC selects the same subroutines as if 'TRIDIAGONAL' had been specified as a subparameter to MATRIX.

Note 7: Generalized Problem Matrix Subparameters.

As discussed earlier, for a problem to be classified "Real Symmetric Generalized", the A and B input matrices must each be symmetric with B positive definite. Unless 'SYMMETRIC' is specified as a subparameter to MATA and both 'SYMMETRIC' and 'POSITIVE DEFINITE' are specified as subparameters to MATB, EISPAC will select a path corresponding to the real non-symmetric generalized case instead (and assume storage of the full A and B matrices).

Note 8: Eigenvalues for the Non-Symmetric Generalized Problem.

The divisions ALPHAR(I)/BETA(I) and ALPHAI(I)/BETA(I) to obtain the real and imaginary parts of the eigenvalues for the real non-symmetric generalized problem are left to the user. The arrays ALPHAR, ALPHAI, and BETA themselves are returned instead, because they have more information than the quotients when B (or both A and B) is nearly singular. Indeed, if B is singular, at least one element of BETA will be zero and division dare not be attempted.

Note 9: Negation and Abbreviation of Alphanumeric Subparameters.

Any alphanumeric subparameter (e.g., 'BALANCE') is negated if its immediately preceding subparameter is 'NO', 'NON', or 'NOT'. Similarly, the subparameters 'YES', 'IS', 'USE', and 'DO' act as identity alphanumeric subparameters. These conventions are useful when you desire to use a single EISPAC call to compute the eigensystem of a matrix which sometimes is to be balanced and sometimes not, say, since the subparameter 'NO' or 'YES' may be an alphanumeric variable.

Any alphanumeric subparameter may be abbreviated by enough of its initial letters to unambiguously distinguish it from all other alphanumeric subparameters. For protection against ambiguities arising from the future introduction of new alphanumeric subparameters, you should use at least three (perhaps four) letter abbreviations when they exist. The alphanumeric subparameters currently in use are listed below (synonyms are separated by commas, antonyms by hyphens):

'DO', 'IS', 'USE', 'YES'
'NO', 'NON', 'NOT'
'REAL' - 'COMPLEX'
'HERMITIAN' - 'GENERAL' (see note 2)
'SYMMETRIC' - 'GENERAL'
'TRIDIAGONAL' - 'FULL'
'POSITIVE DEFINITE'
'NEGATIVE DEFINITE'
'BALANCE'
'ORTHOGONAL' - 'ELEMENTARY'
'UNITARY' - 'ELEMENTARY' (see note 5)
'AX=LBX', 'AZ=WBZ'
'ABX=LX', 'ABZ=WZ'
'BAX=LX', 'BAZ=WZ'
'LARGEST' - 'SMALLEST'.

As a final example illustrating these concepts, suppose you wish to compute the eigenvalues and eigenvectors of an order N real general matrix without balancing and using orthogonal transformations, with retrieval of the error indicator and calls to a user subroutine, in a program with variables dimensioned: A(100,100), WR(100), WI(100), Z(100,100). Then two of several possible calls are:

```
CALL EISPAC (100, N, MATRIX ('REAL', A), VALUES
   (WR, WI), VECTOR (Z), METHOD ('ORTHOGONAL',
   'NO', 'BALANCE'), ERROR (IERROR), SUBR
   (USUB))

CALL EISPAC (100, N, MATRIX ('NOT', 'COMPLEX',
   A), METHOD ('USE', 'ORTHOG', 'NO', 'NO',
   'NO', 'BALAN'), VALUES (WR, WI), VECTOR (Z),
   SUBR (USUB), ERROR (IERROR))
```

(Recall in the last example that keyword parameters can appear in any order in the third parameter position and beyond.)

D. Error Conditions and Returns.

Three types of errors can occur in EISPAC: "decision phase" errors, "execution phase" errors, and "linkage" errors. (See section 3 below.) Decision phase errors arise when EISPAC detects a mistake or inconsistency in the form of the parameters it receives. Execution phase errors are those which it detects during the execution of the eigensystem-package subroutines themselves. Decision phase errors always result in a message describing the error and the further message "EXECUTION CONTINUING" or "EXECUTION TERMINATED" and the appropriate action. Execution phase errors always result in a message describing the error and termination of execution, unless the ERROR keyword was supplied (see below).

Linkage errors may occur either during the decision or execution phases and always cause termination of execution. Linkage error 00 arises when the internal file name EISPACLB has not been associated with a dataset via a DD card or ALLOC command. Linkage error 01 arises when such an association has been made but a requested subroutine or phase of EISPAC cannot be found in the associated dataset. In this case, check that it contains the required eigensystem-package subroutines and the phases of EISPAC. (See sections 3 and 5 below.)

You may modify the behavior of EISPAC when an execution phase error is detected by supplying the ERROR keyword. When it is supplied, its subparameter IERROR is set to a value characterizing the error and return is made to the calling program (i.e., execution is not terminated). Furthermore, you may suppress the printing of the error message by setting IERROR to the value -755 870 989 before calling EISPAC.

A non-zero value of IERROR on return from EISPAC indicates that an execution phase error has occurred. If IERROR is positive, the computation has been abandoned, and few, if any, useful results have been produced. However, if IERROR is negative, the computation has been continued despite the error(s) and some meaningful results have been produced; use of the ERROR keyword thus permits you to recover these results. The values IERROR may assume and their significance are discussed below. (See also (2) and (6).)

Value of IERROR	Error Msg. No.	Significance
0	None	No execution phase errors occurred.
I	00	The calculation of the I-th eigenvalue failed to converge. If the eigensystem of a real or complex general matrix or for a non-symmetric generalized problem was being computed, eigenvalues I+1,I+2,...,N should be correct; otherwise, eigenvalues 1,2,...,I-1 should be correct.
N+I	01	For a real non-symmetric tridiagonal matrix, A(I,1)*A(I-1,3) .LT. 0, violating the restriction discussed above. No useful results are produced.
2*N+I	02	For the eigenvectors of a real non-symmetric tridiagonal matrix, A(I,1) and A(I-1,3) violated the restriction of note 4, above. All eigenvalues, but no eigenvectors, are correct.

3*N+1	03	Either: (1) Parameter MM specified insufficient storage to hold all of the eigenvalues in the interval (RLB,RUB). The only useful result is M, which contains the number of eigenvalues which lie in the interval. Or: (2) It is not possible to compute exactly M eigenvalues (starting from the M11-th) because of exact multiplicity at index M11. No useful results are produced.
3*N+2	04	It is not possible to compute exactly M eigenvalues (starting from the M11-th) because of exact multiplicity at index M11+M-1. No useful results are produced.
4*N+I	None	Not used in EISPAC.
5*N+I	05	The M largest or smallest eigenvalues were being computed by the rational QR method and the I-th eigenvalue failed to converge. All eigenvalues probably have some accuracy, but no stronger statement can be made.
6*N+1	06	The M largest or smallest eigenvalues were being computed by the rational QR method, the matrix was specified negative or positive definite, but the computation suggests it may not be; try again omitting the definiteness specification. No useful results are produced.
7*N+1	07	The eigensystem for a real symmetric generalized problem was being computed and the matrix supplied to MATB was not positive definite. No useful results are produced.
-I	50	The calculation of one or more eigenvectors including the I-th failed to converge. All eigenvalues and the non-zero eigenvectors are correct.

-(N+I)	None	Both error 50 above and error 52 below occurred.
-(2*N+1)	52	Parameter MM specified insufficient columns to hold the selected eigenvectors. All eigenvalues and the first MM columns of (ZR, ZI) or Z are correct.
-(3*N+I)	None	This error predicts that error 02 above will occur. It will not normally be returned from EISPAC but may be observed when the user subroutine is employed.

In addition to the above errors which are diagnosed by EISPAC, other errors may produce System/360-370 completion codes (SCC). The most commonly produced completion codes and their meanings in conjunction with EISPAC are the following:

804 There was not enough storage to dynamically load a requested subroutine. Provide more storage (region), reduce the size of the eigenproblem being solved, or choose a path which requires less temporary storage (see (1) and (6)). The minimum storage requirement for EISPAC is approximately 22,000 (decimal) bytes on System/360-370.

606 Same as SCC804 above.

80A There was not enough storage to allocate a temporary vector or matrix. See suggestions under SCC804 above.

E. Applicability and Restrictions.

EISPAC is applicable to the solution of the standard and generalized eigensystem problems for the classes of matrices for which examples are given above. For further details on applicability, see (6), or consult the "EISPAC Eigensystem Path Chart" (1) to determine which subroutines are used in the path of interest and then the NATS subroutine documents (2) for those subroutines.

3. DISCUSSION OF METHOD AND ALGORITHM.

EISPAC consists of three major parts, the parameter-interpreter phase, the decision phase, and the execution phase, which are co-ordinated by a supervisor. The supervisor employs brief assembly language routines to count the number of subparameters supplied to each of the keywords and the number of parameters supplied to EISPAC itself. It then loads the parameter-interpreter phase dynamically and passes the parameter information to it.

The parameter-interpreter phase analyzes the parameters, checks them for consistency, supplies defaults for omitted optional parameters, and produces a characterization of the problem they specify. It returns this characterization and a standardized list of parameters to the supervisor.

If errors have been detected in the parameter-interpreter phase, the supervisor loads an error message module dynamically which prints error messages and which may terminate execution. Otherwise, the supervisor loads a decision phase appropriate to the problem being solved and passes the characterization to it. The decision phase determines the "path", or sequence, of eigensystem-package subroutines and auxiliary actions (e.g., temporary storage allocations) needed to solve the specified eigenproblem. It then returns a representation of this path to the supervisor.

The supervisor then calls the execution phase, which interprets the representation of the path (and information about order of execution) to load dynamically the required eigensystem subroutines and to allocate and free needed temporary storage. (Loading, allocating, and freeing are performed by operating system requests contained in brief assembly language routines.) When execution is complete, or as soon as a fatal error occurs, the execution phase returns to the supervisor. If execution phase errors have occurred and error messages are to be printed (see section 2.D above), the supervisor loads the error message module dynamically and prints them; execution terminates or continues as described above. When execution continues, the supervisor returns the results of the computation to the calling program.

Certain economies of execution are possible when EISPAC is called repeatedly to solve the same eigenproblem. After the first execution of the path for the problem, usable copies of the eigensystem-package subroutines for it may remain in memory (if sufficient storage is available), and if so they are reused without being reloaded on subsequent executions of the path.

The design and implementation of EISPAC are highly modular, so that additional subroutines and paths can be added relatively easily and with only a minimum of disturbance to existing paths. This modular design of EISPAC and the philosophy behind it are discussed in detail in (7).

4. REFERENCES.

1) Boyle, J.M., Hauser, C.H., and Rothaar, H.R., EISPAC Eigensystem Package Path Chart, Release 2, Argonne National Laboratory, Applied Mathematics Division, February, 1975.

2) NATS Project, Eigensystem Subroutine Package (EISPACK), Subroutines F269 to F298 and F220 to F247.

3) Wilkinson, J.H. and Reinsch, C., Handbook for Automatic Computation, Volume II, Linear Algebra, Springer-Verlag, (1971).

4) Wilkinson, J.H., The Algebraic Eigenvalue Problem, Oxford, Clarendon Press (1965).

5) Rutishauser, H., Interference With an Algol-Procedure, Annual Review in Automatic Programming, Vol. 2 (R. Goodman, Ed.), Pergamon Press, (1961).

6) This publication. (Reference retained to resolve User Guide pointers, thus enabling compatibility with distributed documentation.)

7) Boyle, J.M. and Grau, A.A., Modular Design of a User-Oriented Control Program for EISPACK, Applied Mathematics Division Technical Memorandum No. 242, Argonne National Laboratory, Argonne, Illinois 60439, November 1973.

5. PROGRAM STATISTICS.

A. Additional Entry Point Names in EISPAC.

The entry point names listed below are entry point names in the part of EISPAC loaded with your program. Your program must not include other subroutines or functions having any of these names:

BAND, COREEP, EERMEP, EIDFEP, EISPAC, ERROR, FIDFEP, LINKEP, MATA, MATB, MATRIX, METHOD, SUBR, SUPVEP, VALUES, VECTOR.

In addition, the other phases of EISPAC contain entry points with the names DGNHEP, DGSHEP, DIDFEP, DSNHEP, EGNHEP, EGSHEP, ERMSEP, ERSEP, ESNHEP, FIDFEP, IBCMEP, IBCOM#, ICHREP, INTREP, and LKUPEP. These entry points will not, however, conflict with those in your program, since these phases are loaded dynamically.

B. Common Storage.

None.

C. Space Requirements.

The parts of EISPAC loaded with your program (subroutine EISPAC and the supervisor) require 2738 (HEX) = 10040 (DECIMAL) bytes under FORTRAN H, OPT=2, Release 21.7 of OS/360. However, EISPAC requires an additional amount of storage for the dynamically loaded subroutines and the temporary arrays some of them require. At a minimum, this requirement is 2EA0 (HEX) = 11936 (DECIMAL) bytes for the parameter-interpreter phase of EISPAC. During execution, up to 3100 (HEX) = 12544 (DECIMAL) bytes are required for the appropriate execution phase of EISPAC and a typical EISPACK subroutine, plus whatever additional storage is needed for dynamically allocated temporary arrays. See (1) and (2) for further information for estimating this dynamically-determined amount of storage.

6. CHECKOUT.

A. Test Cases.

EISPAC has been tested on the same matrices as have the separate EISPACK subroutines. See Section 3 -- "VALIDATION OF EISPACK".

B. Accuracy.

The accuracy of EISPAC depends directly upon the subroutines used from the EISPACK package (2). Consult the path chart (1) to learn which subroutines have been selected, and then reference the separate accuracy statements that apply to these subroutines.